ESD-TR-84-190

MTR-9420

DESIGN GUIDELINES FOR USER-SYSTEM INTERFACE SOFTWARE

By

SIDNEY L. SMITH
JANE N. MOSIER

SEPTEMBER 1984

Prepared for

DEPUTY FOR ACQUISITION LOGISTICS AND TECHNICAL OPERATIONS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Massachusetts

Project No. 522A
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract No. F19628-84-C-0001

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

N. ANN KUO, 2LT, USAF
Project Manager,
Requirements Analysis

WILLIAM J. LETENDRE
Program Manager,
Computer Resource Management
Technology

FOR THE COMMANDER

ROBERT J. KENT
Director, Computer Systems Engineering
Deputy for Acquisition Logistics
and Technical Operations

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MTR-9420 ESD-TR-84-190	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION The MITRE Corporation 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Burlington Road Bedford, MA 01730	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Deputy for Acquisition Logistics & Tech. Ops. 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-84-C-0001
8c. ADDRESS (City, State and ZIP Code) Electronic Systems Division, AFSC Hanscom AFB, MA 01731-5000	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. 522A / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) DESIGN GUIDELINES FOR USER-SYSTEM INTERFACE SOFTWARE	
12. PERSONAL AUTHOR(S) Sidney L. Smith, Jane N. Mosier	
13a. TYPE OF REPORT Final Report 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1984 September 15. PAGE COUNT 460
16. SUPPLEMENTARY NOTATION	

17 COSATI CODES		
FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
COMPUTER-BASED SYSTEMS
DESIGN GUIDELINES
HUMAN FACTORS
INFORMATION SYSTEMS
USER-SYSTEM INTERFACE

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report offers guidelines for design of user interface software in six functional areas: data entry, data display, sequence control, user guidance, data transmission, and data protection. This report revises and extends previous compilations of design guidelines (cf. Smith, 1982b; Smith and Aucella, 1983a). Efforts will continue next year to improve these guidelines still further.

If you are a teacher, a student, a human factors practitioner or researcher, these guidelines can serve as a starting point for the development and application of expert knowledge. But that is not the primary objective of this compilation. The guidelines are proposed here as a potential tool for designers of user interface software.

If you are a system analyst, you can review these guidelines to establish design requirements. If you are a software designer, you can consult these guidelines to (cont.)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☒ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Susan R. Gilbert	22b. TELEPHONE NUMBER (Include Area Code) (617) 271-8088 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

19. (Concluded)

derive the specific design rules appropriate for your particular system application. That translation from general guidelines to specific rules will help focus attention on critical user interface design questions early in the design process.

If you are a manager responsible for user interface software design, you may find in these guidelines a means to make the design process more efficient. Guidelines offer a means to establish rules for coordinating individual design contributions, a means to make design decisions just once rather than leaving them to be made over and over again by individual designers, a means to define detailed design requirements and to evaluate user interface software in comparison with those requirements.

The design of user interface software will often involve a considerable investment of time and effort. Design guidelines can help ensure the value of that investment.

SUMMARY

This report offers guidelines for design of user interface software in six functional areas: data entry, data display, sequence control, user guidance, data transmission, and data protection. This report revises and extends previous compilations of design guidelines (cf. Smith, 1982b; Smith and Aucella, 1983a). Efforts will continue next year to improve these guidelines still further.

If you are a teacher, a student, a human factors practitioner or researcher, these guidelines can serve as a starting point for the development and application of expert knowledge. But that is not the primary objective of this compilation. The guidelines are proposed here as a potential tool for designers of user interface software.

If you are a system analyst, you can review these guidelines to establish design requirements. If you are a software designer, you can consult these guidelines to derive the specific design rules appropriate for your particular system application. That translation from general guidelines to specific rules will help focus attention on critical user interface design questions early in the design process.

If you are a manager responsible for user interface software design, you may find in these guidelines a means to make the design process more efficient. Guidelines offer a means to establish rules for coordinating individual design contributions, a means to make design decisions just once rather than leaving them to be made over and over again by individual designers, a means to define detailed design requirements and to evaluate user interface software in comparison with those requirements.

The design of user interface software will often involve a considerable investment of time and effort. Design guidelines can help ensure the value of that investment.

ACKNOWLEDGMENT

This report was prepared by The MITRE Corporation. The work reported here was sponsored by the Computer Engineering Applications Division, Deputy for Acquisition Logistics and Technical Operations of the Electronic Systems Division (ESD) of the United States Air Force Systems Command, Hanscom Air Force Base, MA 01731. Funding for this work was provided by the Air Force Computer Resource Management Technology Program, Program Element 64740F, under ESD/MITRE Project 5220.

The Computer Resource Management Technology Program supports development and transition into active use of tools and techniques needed to cope with the explosive growth in Air Force systems that use computer resources. The objectives of that Program are:

to provide for the transition to Air Force systems of computer system developments in laboratories, industry, and academia;

to develop and apply software acquisition management techniques to reduce life cycle costs;

to provide improved software design tools;

to address problems associated with computer security;

to develop advanced software engineering tools, techniques, and systems;

to support the implementation of high-order programming languages, e.g., Ada;

to improve human engineering of computer systems; and

to develop and apply computer simulation techniques in support of system acquisition.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Concluded)

INTRODUCTION

In computer-based information systems, special attention must be given to design of the user-system interface (USI) software. Guidelines for USI software design have been compiled as a continuing effort, sponsored by the Air Force Electronic Systems Division (ESD). Four previous ESD reports have dealt with this subject (Smith, 1980; 1981a; 1982b; Smith and Aucella, 1983a).

This present report revises and expands previously published material, and proposes a more comprehensive set of guidelines for design of USI software in computer-based information systems. Although a great many changes have been made, much of the text and guidelines material in this report will seem familiar to the readers of previous reports.

Different people will read this report for different reasons -- teachers and students, human factors practitioners and researchers, system analysts and software designers, and their managers. Each reader will bring to the task a different background of experience and interests. Thus some introductory comments are needed to help familiarize readers with the general problems of USI design and the particular need for guidelines to design USI software.

For the skeptical reader, this introduction offers arguments in favor of guidelines for USI software design. For the enthusiast who may imagine that guidelines can solve all design problems, this introduction will note some of their limitations. For those readers who wish to apply design guidelines, this introduction concludes with a description of how this report is formatted, how the guidelines are presented here and how they are annotated, and some recommendations for how the guidelines material should be used.

INFORMATION SYSTEMS

Discussion of information systems is couched in general terms throughout this report, because USI design problems must be resolved in many different kinds of systems. Guidelines for USI software design are proposed here for use in Air Force command, control, and communication systems; but these guidelines should prove useful in other information system applications as well.

Current use of computers covers a broad range of applications. At one extreme are large, general-purpose computer installations, used by different people for different purposes. Users are expected to provide exact instructions (a computer program) in order to

request data processing. Many users have programming skills. All users must have some knowledge of the system and its capabilities.

At the other extreme are the small, special-purpose computers used as components within larger systems. Such component computers may regulate the ignition of an automobile, or the tuning of a television set. Their purpose is to help make system operation easier. Users do not interact directly with component computers, and indeed may not even know they are there.

Between these extremes of general-purpose installations and component computers, there are a variety of applications in which computers are used to support what are commonly called information systems. Information systems are task-oriented rather than general-purpose, being designed to help perform defined jobs.

Applications of information systems range from relatively simple data entry and retrieval (e.g., airline reservations) through more complex monitoring and control tasks (inventory control, process control, air traffic control), to jobs requiring long-term analysis and planning. Military information systems fall within this broad range from simple to complex.

The users of information systems must interact with a computer in some explicit fashion to accomplish the information handling tasks needed to get their jobs done. These users differ in ability, training and job experience. They may be keenly concerned with task performance, but probably have little knowledge of (or interest in) the computers themselves. Design of the user-system interface must take these human factors into account.

Information systems are designed for workers, and not for hobbyists. A personal computer for use at home would not in this sense be considered an information system, but is simply a smaller version of a general-purpose computer. Of course, it sometimes happens that application software and overlay software must be designed for general-purpose computers, in order to help users perform particular tasks. Such software will require care in design of the user interface.

USER-SYSTEM INTERFACE (USI)

What is the user-system interface? In accord with recommended usage, here the phrase is defined broadly to include all aspects of system design that affect a user's participation in information handling transactions (Smith, 1982a).

Some people may think of the user interface primarily in terms of its directly observable aspects, i.e., the physical work environment and the hardware facilities at the user's work station. Such physical aspects, sometimes called the man-machine interface, have been the subject of conventional human engineering study. There the concern is for illumination, seating, workplace arrangement, keyboard layout, display contrast, symbol size, etc.

Good design of the physical workplace is important, of course, but by itself is not sufficient to ensure effective job performance. Also important are less tangible aspects of information system design -- the ways in which data are stored and manipulated, including paper files and forms, if any, and the procedures and processing logic that govern data handling transactions. Forms, procedures and logic involve software design, the design of computer programs to permit hardware and paper to be used in conjunction with automated data processing.

If the user interface is conceived in these broad terms to encompass all factors influencing user-system interaction, then it is obvious that good USI design will be critical for effective system operation. In any automated information system, whether its work stations are used for data input, calculation, planning, management or control, USI design will have a significant effect on system performance. For good USI design, the analysis of information handling tasks, definition of operating procedures, equipment selection, workspace configuration, and especially the design of USI software -- all must be handled with care.

USI SOFTWARE

The significant role of USI software in system design poses a special challenge to human factors practitioners, recognized early by Parsons:

> . . . what sets data processing systems apart as a special breed? The function of each switch button, the functional arrangement among the buttons, the size and distribution of elements within a display are established not in the design of the equipment but in how the computer is programmed. Of even more consequence, the 'design' in the programs establishes the contents of processed data available to the operator and the visual relationships among the data. In combination with or in place of hardware, it can also establish the sequence of actions which the operator must use and the feedback to the operator concerning those actions.
>
> (1970, page 169)

with timely feedback so that a user can correct errors while the data are still fresh in mind and/or while documented source data are still at hand. Here the computer should preserve the context of each data entry transaction, saving correct items so that the user does not have to enter those again while changing incorrect items.

Preservation of context is, of course, important in all aspects of user-system interaction, with implications for data display, sequence control and user guidance, as well as for data entry. The importance of context is emphasized again in the discussion of those other functional areas.

Another important design concept is flexibility. The idea is often expressed that interface design should adapt flexibly to user needs. But the specific means of achieving such flexibility must be spelled out in user interface design guidelines. For data entry functions it is important that the pacing of inputs be controlled flexibly by the user. Tasks where the pacing of user inputs is set by a machine (for example, keying ZIP codes at an "automated" post office) are stressful and error-prone.

Aside from flexibility in pacing, users will often benefit from having some flexible choice in the ordering of inputs. What is needed for interface design is some sort of suspense file(s) to permit flexible ordering of data entries, including temporary omission of unknown items, backup to correct mistaken entries, cancellation of incomplete transactions, etc.

As noted above, users may also benefit from flexibility in defining default options to simplify data entry during a sequence of transactions. Some systems include only those defaults anticipated by the designers, which may not prove helpful to the user in a particular instance. Thus the concept of flexibility is related to the need to establish and maintain context, and related also to many other aspects of interface design. How can all of these related ideas be separated to establish a useful structure for organizing design guidelines?

The basic organization adopted here for data entry guidelines is in terms of function. Guidelines are presented for different kinds of data entry and for different kinds of computer processing support. Some topics, such as "abbreviation", seem to pertain to all data entry functions, and are included here in an initial section dealing generally with the subject. To help readers become familiar with the topical organization, and to find particular material, the guidelines that follow are introduced by an annotated table of contents for this section.

influence data entry tasks, as suggested by current advocacy of voice input.

But the major need in today's information systems is for improving the logic of data entry, and it is there that design guidance should prove most helpful. Thus the guidelines presented here deal with data entry functions, insofar as possible, without regard to their hardware implementation. As an example, guidelines for position designation attempt to deal with pointing as a generic function, rather than providing specific rules for pointing with different kinds of devices.

The general objectives of designing data entry functions are to establish consistency of data entry transactions, minimize input actions and memory load on the user, ensure compatibility of data entry with data display, and to provide flexibility of user control of data entry. Stated in such general terms, these principles do not provide helpful guidance to designers. Somehow these general ideas must be converted into more specific guidelines.

The process of converting general principles into specific rules will lead to a considerable proliferation of ideas. With regard to minimizing input actions, one specific rule might be that a user should not have to enter the same data twice. Probably every designer knows this, even if it is a rule that is sometimes forgotten. A related rule might be that a user should not have to enter data already entered by another user. That seems to make good sense, although one could imagine occasional exceptions when cross validation of data inputs is required.

How can duplicative data entry be avoided in practice? The solution lies in designing the user interface (programming the computer) to maintain context. Thus when a user identifies a data category of interest, say a squadron of aircraft, the computer should be able to access all previously entered data relevant to that squadron and not require the user to enter such data again.

In repetitive data entry transactions the user should have some means of establishing context. One method is to allow users to define default entries for selected data items, in effect telling the computer that those items will stay the same until the default value is changed or removed. If a user enters one item of data about a particular squadron, it should be possible to enter other items thereafter without having to re-identify that squadron.

Context should also be preserved to help speed correction of input errors. One significant advantage of on-line data entry is the opportunity for immediate computer validation of user inputs,

SECTION 1

DATA ENTRY

Data entry refers to user actions involving input of data to a computer, and computer responses to such inputs. The simplest kind of data entry consists merely of pointing at something -- selecting an item (or designating a position) on a computer-generated display. In more complicated modes of data entry, a user may have to control the format of data inputs as well as their contents. Thus questions of format control in text entry/editing, and in graphic interaction, may properly be considered questions of data entry.

However, user inputs that initiate or interrupt transactions -- i.e., command entries, option selections from a displayed menu, activation of function keys -- pose rather different questions of design. Such user inputs are discussed in a later section concerned with sequence control.

Data can be entered into a computer in a variety of different ways. Users can designate position or direction by pointing at a display. Users can enter numbers, letters, or more extended textual material by keyed inputs, or in some applications by spoken inputs. Data may be keyed into displayed forms or tables, into constrained message formats or as free text. In graphic interaction users may draw pictures or manipulate defined graphic elements. These different types of data entry all deserve consideration here.

The computer will also play a role in the data entry process, guiding users who need help, checking data entries to detect errors, and providing other kinds of data processing aids. The designer of user interface software must be concerned about computer processing logic as well as display formats and input devices.

Data entry is heavily emphasized in clerical jobs, and many other jobs involve data entry to some degree. Because data entry is so common, and because inefficiencies caused by poorly designed data entry transactions are so apparent, most of the published recommendations for good user interface design deal extensively with data entry questions. Human factors specialists can probably give better advice about data entry functions than in any other functional area of user interface design.

Data entry transactions are necessarily influenced by hardware selection, and the proper design of input devices has received considerable attention, including concern for standardization of keyboard layouts. Future advances in hardware design may well

FUTURE GUIDELINES DEVELOPMENT

Future efforts will be directed toward revising and expanding still further the USI design guidelines presented here. Critical review is needed. Design guidelines are based on judgment, and so a broad range of judgment is needed to assess them. Improvement of design guidelines will require a collaborative effort with other people working on user interface design.

Readers are encouraged to recommend changes to the guidelines proposed here. Do some of the guidelines seem wrong? Are some unclear? Can you suggest additions or changes to the guidelines, better wording, further examples, exceptions, references to supporting data? If so, please use the suggestion sheet that is appended as the last page in this report.

In addition to revision and expansion of the guidelines, it is necessary to evaluate their usefulness in practical application to user interface design. Application of USI design guidelines is presently being explored in Air Force system acquisition programs. If you find an opportunity to apply these guidelines in your own work, please contact the authors of this report so that we can share information.

Design rules can be derived from the guidelines material, but that conversion from guidelines to rules should be performed as an integral part of the design process, serving to focus attention on critical design issues and to establish specific design requirements. Once agreed design rules are established, those rules can be maintained and enforced by the managers of system development programs.

Specific design rules probably cannot be imposed effectively at the outset of system development, by some external agency -- by a sponsoring organization or by a marketing group. It is the process of establishing design rules that should be imposed, rather than the rules themselves. A software design contractor might reasonably be required to establish rules for the design of user interface software, subject to review by the contracting agency. And available guidelines could be cited as a potentially useful reference for that purpose.

Some other cautionary comments about the application of guidelines deserve consideration here. Guidelines cannot take the place of experience. An experienced designer, one skilled in the art, might do well without any guidelines. An inexperienced designer might do poorly even with guidelines. Few designers will find time to read an entire book of guidelines. If they do, they will find it difficult to digest and remember all of the material. If guidelines are to be helpful, and/or the rules derived from guidelines, must be kept continually available for ready reference.

Guidelines cannot take the place of expert design consultants, or at least not entirely. A design expert will know more about a specific topic than can be presented in the guidelines. An expert will know what questions to ask, as well as many of the answers. An expert will know how to adapt generally stated guidelines to the specific needs of a particular system design application. An expert will know how to trade off the competing demands of different guidelines, in terms of operational requirements.

Guidelines cannot replace task analysis. Indeed, many of the guidelines here, considered along with their associated commentary, imply the need for careful task analysis to determine design requirements. Guidelines will not necessarily save work in user interface design, but in fact may entail extra work, at least in the initial stage of establishing design rules. If that initial work is well done, however, then subsequent software design should be more efficient and, of course, should produce a better user interface.

In selecting guidelines for use, the functional organization of material in this report can aid association of guidelines with required user interface functions. As a designer considers each required function for a particular system application, pertinent guidelines can be readily referenced. Conversely, if a function is not required, designers can readily identify those corresponding design guidelines which will not be relevant to functional requirements.

In establishing rules for USI software design, it must be recognized that software design will necessarily be dependent in some degree on the means chosen for hardware implementation of the user interface. That is acknowledged in the contingent wording of some of the guidelines proposed here. And certainly there are hardware features implied in some of these guidelines, including function keys for ENTER, CONFIRM, BACKUP, CANCEL, DITTO, PRINT, and various tab keys for controlling cursor position in USI designs involving form-filling dialogues. For the most part, however, the guidelines are stated in terms of software function rather than hardware specification. These guidelines could be amplified by system developers to include more explicit specification of hardware features, where that seems appropriate.

When agreed design rules have been established, it may be useful to assign different weights to the various rules, indicating which are more important than others. Such weighting will help resolve the trade-offs that are an inevitable part of the design process. Weighting of desired features will also provide a basis for subsequent evaluation of proposed (or accomplished) user interface designs.

Once agreed design rules have been established, they should be documented for reference by software designers and others involved in system development. Documentation of agreed rules, subject to periodic review and revision as necessary, will help coordinate the design process. Documented rules can then be applied consistently for a given application. With appropriate modifications, rules adopted for one application might later be used for other applications in other information systems.

If guidelines are applied in the way described here, there are some significant implications for the role of guidelines in system development. Generally stated guidelines should be offered to designers as a potential resource, rather than imposed contractually. It is only specifically worded design rules that can be enforced, not guidelines.

APPLYING THE GUIDELINES

If guidelines are proposed for application in a variety of systems, then they must be written in general terms. A guideline that said every display should have a capitalized title centered in the second line would be too restrictive. Such a guideline would constitute a specific design rule that might be satisfactory in some applications but not in others. In this example, the guideline could be rephrased in more general terms to say that every display should be consistently identified in some distinctive way. Stated in those general terms, the guideline would have much broader application.

But how can a designer apply a generally stated guideline? Continuing the example above, there are many ways to identify displays consistently and distinctively. A designer might prefer simply to be told which way to do it, rather than having to decide the question anew for each application. Designers may be disappointed if they expect guidelines to state specific rules, but find only general advice instead.

How can we resolve this conflict between the need to phrase guidelines generally for broad application and the designer's need for specific rules? In brief, the general guidelines must be translated or converted into specific rules that the designer can follow. Application of guidelines will thus involve questions of how they should be converted into design rules, who should do it, and when.

Considering these questions in reverse order, specific rules should be established early in the design process, before any actual design of user interface software. The establishment of design rules should be a joint responsibility of system analysts assessing design requirements, software designers assessing feasibility, and their managers. It may also be helpful to include representatives of the intended system users in this process, to ensure that proposed design features will meet operational requirements.

Establishment of design rules might begin with review of the guidelines material. Certain guidelines might be discarded as being irrelevant to the particular system application at hand. Other guidelines might be modified. All guidelines that are accepted for use should be reworded as necessary to convert them into agreed design rules. In that conversion, some guidelines might be considerably expanded. For example, a guideline that says displays should be consistently formatted might be converted into a series of eight or ten rules specifying the exact format to be adopted for different elements in a display.

illustrate the guidelines, and are not intended to limit the interpretation of guidelines.

Where the validity of a guideline is contingent upon special circumstances, examples may be followed by noted exceptions. Those exceptions are intended to limit the interpretation of a guideline.

Where further clarification of a guideline seems needed, examples and exceptions may be followed by supplementary comments. Those comments may explain the reasoning behind a guideline, or suggest possible ways to interpret the guideline, or note relations between one guideline and another.

Where a guideline has been derived from (or is related to) a particular published source, a reference note is added citing author(s) and date. Complete citations for those references are listed following Section 6 of the guidelines. Where a guideline corresponds with other published design standards or guidelines, which is often the case, reference citations are given by letter codes. Those codes are explained in the reference list.

Where a guideline is specifically related to other guidelines, appropriate cross references are given. Those cross references permit an interested reader to explore how a particular topic is dealt with in different sections of the guidelines.

Toward the back of this report, following the guidelines is the reference list. Following the reference list there is a list of the titles for all 679 guidelines, which may help a reader who is trying to find guidelines that pertain to a particular topic.

Following the list of guideline titles, there is a glossary. The glossary defines word usage in the guidelines, for those words that are used here differently or more narrowly than in the general literature on USI design. There is no question that we need more consistent terminology in this field.

Following the glossary, and concluding this report, there is a topical index of the guidelines material. That index is intended to help readers find guidelines on a particular subject, independently of the functional organization that has been imposed on the guidelines material.

These notes on organization and format should serve to allow a student of the subject to skim the guidelines material and find information on different topics. For those readers who seek to apply the guidelines in software design, some further comments are needed.

The overall organization of guidelines follows the structure established in a checklist of functional capabilities that has been developed for USI requirements definition (Smith, 1984). Each section of guidelines covers a different functional area of user-system interaction, although there is necessarily some overlap in topical coverage from one section to another. Within each section, guidelines are grouped by specific functions. Each function has its own numeric designator, as listed in the table of contents for this report.

Each section begins with an introductory discussion of design issues relating to the general functional area. That discussion provides some perspective for the guidelines that follow. The discussion concludes with brief definitions of the various user interface functions covered in that section of the guidelines, along with an internal table of contents for that section, which may help to lead a reader directly to those functions of immediate interest.

Function definitions are repeated in boxed format to begin the listing of guidelines under each function. Those definitions should aid reader understanding of the material, and the boxed format will provide a notable visual indicator that a new series of guidelines has begun.

The guidelines themselves are numbered sequentially under each function, in order to permit convenient referencing. Under any function there will usually be guidelines pertaining to various subordinate topics. Each guideline has been given a short title to indicate its particular subject matter. Each guideline that introduces a new topic is marked with a black dot (•) before the guideline title. When succeeding guidelines deal with the same topic, each is marked with a white dot (o) to imply that it should be interpreted in conjunction with other related guidelines.

Following its number and title, each guideline is stated as a single sentence. Guidelines are worded as simply as possible, usually in general terms to permit broad application, but sometimes with contingent phrasing intended to define a more limited scope of application.

In many instances, a stated guideline will be illustrated by one or more examples. Specific examples can help clarify a generally worded guideline. Sometimes a reader will say, "I didn't really understand the guideline until I saw the example." But there is a potential hazard in examples. Because any example must be narrowly specific, a reader who relies on that example may interpret the guideline as having a narrower meaning than was intended. It is important to recognize that examples are presented here to

In a survey of people concerned with USI design (Smith and Mosier, 1984), respondents generally support Penniman's activist position. Given a choice between trying to develop a complete set of USI guidelines now, when many of them must be based on judgment rather than experimental data, or else accepting only a partial set of guidelines based on evaluated research, most respondents would go with judgment now.

It is clear, of course, that system developers cannot wait for future research data in making present design decisions. To meet current needs, several in-house handbooks have been published to guide USI design within particular organizations (NASA, 1979; Galitz, 1980; Parrish, Gates, Munger, Grimma, and Smith, 1982; Brown, Brown, Burkleo, Mangelsdorf, Olsen, and Perkins, 1983).

These in-house guidelines draw heavily from those in earlier publications, especially the influential IBM report by Engel and Granda (1975), as modified by the authors' own good judgment. They will help system developers until such time as a comprehensive USI design standard becomes available.

The ESD/MITRE compilation of USI design guidelines over the past several years has been built on the work of our predecessors, and will help support the work of others to follow. Each year our compilation has grown larger. Two years ago we had compiled 375 guidelines (Smith, 1982b). Last year's report contained some 580 guidelines. In this present report there are 679. Next year's report will include even more.

This present compilation, although still not complete, represents the most comprehensive published guidance available for USI software design, and for that reason is recommended as a reference in current system acquisition programs.

GUIDELINES ORGANIZATION

In the numbered sections of this report, guidelines have been organized within six functional areas of user-system interaction:

Section	USI Functions	Number of Guidelines
1	Data Entry	143
2	Data Display	163
3	Sequence Control	168
4	User Guidance	97
5	Data Transmission	40
6	Data Protection	68

version. That version included nine pages dealing with USI software design, in a section titled "Personnel-Computer Interface". That material was later expanded to 19 pages, titled "User-Computer Interface", in a revision of MIL-STD-1472C (1983). Thus a beginning has been made, but much more is needed. The question is, how can needed guidance for USI software design be developed?

USI DESIGN GUIDELINES

Until several years ago, there had been no serious attempt to integrate the scattered papers, articles and technical reports that constitute the literature of user-computer interaction. A first step was made, under sponsorship of the Office of Naval Research (ONR), in compilation of an extensive bibliography on this subject (Ramsey, Atwood and Kirshbaum, 1978). A significant follow-on effort culminated in publication by Ramsey and Atwood (1979) of a comprehensive summary of this literature.

In reviewing the literature, it becomes apparent that most published reports dealing with the user-computer interface describe applications rather than design principles. A popular early book on the design of user-computer dialogues offered stimulating examples, covering a range of on-line applications, but was disappointing in its failure to emphasize design principles (Martin, 1973). The ONR bibliography cited above includes 564 items, but identifies only 17 as offering design guidelines.

Although accepted principles for USI design have not been available, some work has been accomplished toward that end. As increasing experience has been gained in the use of on-line computer systems, some experts have attempted to set forth principles ("guidelines", "ground rules", "rules of thumb") for design of the user-computer interface. That work has not produced a comprehensive set of guidelines, but it does offer a foundation on which to build. In aggregate, the evidence of concern for USI design principles is encouraging, but there is still much to be learned.

Military agencies, of course, are not the only organizations seeking guidelines for USI design. There is increasing interest in this topic within industrial and commercial organizations, and throughout the general community of people who develop and use information systems. David Penniman, writing for the User On-Line Interaction Group of the American Society for Information Sciences, has cited the need for "an interim set of guidelines for user interface design based on available literature and pending the development of better guidelines as our knowledge increases" (1979, page 2). Penniman goes on to remind us that interim guidelines are better than no guidelines at all.

user interface software -- including selection and formatting of displayed data, consistency in wording and procedures, on-line user guidance, explicit error messages, re-entry rather than overtyping for data change, elimination of abbreviations, etc. -- resulted in significantly improved system performance. Data entry was accomplished 25 percent faster, and with 25 percent fewer errors. How can this kind of design improvement be achieved in general practice?

DESIGN PRACTICE

Looking at current design practice, it seems fair to characterize USI software design as art rather than science, depending more upon individual judgment than systematic application of knowledge (Ramsey and Atwood, 1979; 1980). As an art, USI design is best practiced by experts, by specialists experienced in the human engineering of computer systems. But such experts are not always available to help guide system development, and it is clear that they cannot personally guide every step of USI design. What is needed is some way to embody expert judgment in the form of explicit guidelines for USI design.

For military information systems, Military Specification MIL-H-48655B (1979) calls for a system development sequence starting with requirements analysis, functional specification and design verification. The actual course of USI software design will sometimes depart from that desired sequence. There may be no explicit attempt to determine USI requirements. Specifications may include only rudimentary references to USI design, with general statements that the system must be "easy to use". In the absence of effective guidance, both the design and implementation of USI software may become the responsibility of programmers unfamiliar with operational requirements. Detection and correction of design flaws may occur only after system prototyping, when software changes are difficult to make.

Human engineering standards and design handbooks have in the past been of little use to the software designer. The recently published human factors design handbook by Woodson (1981) is typical. Its nearly 1000 pages include only three pages of general material on information processing, and there is no reference to computer systems in its index.

MIL-STD-1472B (1974), for many years the major human engineering design standard for military system procurement, was concerned almost exclusively with hardware design and physical safety. In 1981, MIL-STD-1472 was published in a revised "C"

In a constrained environment, such as that of many military and commercial information systems, users may have little choice but to make do with whatever interface design is provided. There the symptoms of poor USI design may appear in degraded performance. Frequent and/or serious errors in data handling may result from confusing USI design. Tedious user procedures may slow data processing, resulting in longer queues at the checkout counter, the teller's window, the visa office, the truck dock, or any other workplace where the potential benefits of computer support are outweighed by an unintended increase in human effort.

In situations where degradation in system performance is not so easily measured, symptoms of poor USI design may appear as user complaints. The system may be described as hard to learn, or clumsy, tiring and slow to use. The users' view of a system is conditioned chiefly by experience with its interface. If USI design is unsatisfactory, the users' view of the system will be negative, regardless of any niceties of internal computer processing.

USI design deficiencies can arise from problems in system development. Different portions of the user interface may be designed by different people, who share no common view of desired operational procedures. The result is design inconsistency. When users must handle different tasks differently, or even different transactions within the same task differently, an unnecessary burden is imposed on learning and use of the system.

In some applications, system developers must worry about design inconsistencies among different systems. Within a large industrial firm, or within a government agency or a military service, many different information systems may be developed for different purposes in different parts of the organization. Examples of inter-system inconsistencies of USI design have been documented in a study of battlefield information systems, sponsored by the Army Research Institute (Sidorsky and Parrish, 1980).

Where USI design is established independently for each system, the result can be to impose a burden of incompatible habits on the user who must move from one system to another. Conversely, some standardization of USI design across systems might reduce user learning problems and improve user performance. And that is the encouraging side to this picture. If deficiencies in USI design can degrade performance, it is equally true that improvements to USI design can produce better performance.

A convincing demonstration of design improvement has been reported by Keister and Gallaway (1983). Those authors describe a data entry application in which relatively simple improvements to

Continuing concern for USI software design is suggested by phrases such as "software psychology" (cf. Shneiderman, 1980). But USI design cannot be the concern only of the psychologist or the human factors specialist. It is a significant part of information system design that must engage the attention of system developers, designers, and ultimately system users as well.

Not only is USI software design critical to system performance, it can also represent a sizable investment of programming effort during initial system development, and during subsequent system modification to accommodate changing operational requirements (so-called "software maintenance"). Just how sizable is that investment? The answer, of course, will depend upon the particular type of information system that is being considered.

In a recent survey (Smith and Mosier, 1984), people involved in the design of information systems were asked to estimate the percent of operational software devoted to implementing the user interface. Overall, the average estimate was that USI design comprises 30 to 35 percent of operational software. Estimates for individual systems ranged from 3 to 100 percent, reflecting the fact that some computer systems require a much higher investment in USI design than others, depending upon their purpose.

Because of the critical importance of USI software, the bulk of this report and the guidelines proposed here will deal almost exclusively with problems of user interface software design.

DESIGN SIGNIFICANCE

Given the broad definition of the user interface adopted here, it is obvious that deficiencies in USI design can result in degraded system performance. To be sure, users can sometimes compensate for poor design with extra effort. Probably no single USI design flaw, in itself, will cause system failure. But there is a limit to how well users can adapt to a poorly designed interface. As one deficiency is added to another, the cumulative negative effects may eventually result in system failure, poor performance, and/or user complaints.

Outright system failure can be seen in abandoned systems, or decreased system use where use is optional. There may be retention of (or reversion to) manual data handling procedures, with little use of automated capabilities. When a system fails in this way, the result is disrupted operation, wasted time, effort and money, and failure to achieve the potential benefits of automated information handling.

- Data entry refers to user actions involving input of data to a computer, and computer responses to such inputs.
- Position designation refers to user selection and entry of a position on a display, or of a displayed item.
- Direction designation refers to user entry of directional data (azimuth, bearing, heading, etc.) on a display.
- Text entry refers to the initial entry and subsequent editing of textual material, including messages.
- Data forms permit entry of predefined items into labeled fields of specially formatted displays.
- Tables permit data entry and display in row-column format, facilitating comparison of related data sets.
- Interactive graphics permit convenient entry and change of data representing spatial or other functional relations.
- Data validation refers to checking entries for incorrect content and/or format, as defined by software logic.
- Other kinds of computer processing may be provided to facilitate data entry.
- Changes to software design of data entry functions may be needed to meet changing operational requirements.

Objectives:

Consistency of data entry transactions
Minimal entry actions by user
Minimal memory load on user
Compatibility of data entry with data display
Flexibility for user control of data entry

Guidelines:

Data entry refers to user actions involving input of data to a computer, and computer responses to such inputs.

-1 • Entry via Primary Display -1

When data entry is a significant part of a user's task, entered data should appear on the user's primary display.

Example: Entry via typewriter is acceptable only if the typewriter itself, under computer control, is the primary display medium.

Comment: When the primary display is basically formatted for other purposes, such as a graphic display for process control, a separate window on the display may have to be reserved for data entry.

-2 • Feedback During Data Entry -2

Provide displayed feedback for all user actions during data entry; display keyed entries stroke by stroke.

Exception: For reasons of data protection, it may not be desirable to display passwords and other secure entries.

Reference: EG 6.3.7; MS 5.15.2.1.2, 5.15.2.2.3.

See also: 1.0-11, 1.0-12, 3.0-11, 4.2-1.

-3 o Fast Response -3

The computer should acknowledge data entry actions rapidly, so that users are not slowed or paced by delays in computer response; for normal operation, delays in displayed feedback should not exceed 0.2 seconds.

Example: A key press should be followed by seemingly immediate display of its associated symbol, or by some other appropriate display change.

Comment: This recommendation is intended to ensure efficient operation in routine, repetitive data entry tasks. Longer delays may be tolerable in special circumstances, perhaps to reduce variability in computer response, or perhaps in cases where data entry comprises a relatively small portion of the user's task.

Comment: Note that this guideline refers to acknowledgment rather than final processing of entries, where processing may be deferred pending an explicit ENTER action. If displayed acknowledgment of entries is unavoidably slow, then it may be desirable to impose a corresponding delay on the subsequent ENTER action.

Reference: EG Table 2.

See also: 3.0-15, 3.0-16.

-4 • Single Method for Entering Data -4

Design the data entry transactions and associated displays so that a user can stay with one method of entry, and not have to shift to another.

Example: Minimize shifts from lightpen to keyboard entry and then back again.

Example: A user should not have to shift from one keyboard to another, or move from one work station to another, to accomplish different data entry tasks.

Comment: This, like other guidelines here, assumes a task-oriented user, busy or even overloaded, who needs efficiency of data entry.

Reference: BB 2.11; EG 6.1.1; Foley and Wallace, 1974.

See also: 1.1-14.

-5 • Defined Display Areas for Data Entry -5

Where data entry on an electronic display is permitted only in certain areas, as in form filling, the display format should provide clear visual definition of the entry fields.

Example: Data entry fields might be underlined, or perhaps highlighted by reverse video.

Exception: For general text entry of variable (unrestricted) length, no field delimiters are needed. In effect, keyed text entries can replace nothing (null characters).

Comment: Display formats with field delimiters provide explicit user guidance as to the location and extent of data entry fields. Where delimiters extend throughout an entry field, as in underlining, then any keyed data entries should replace the delimiter characters on the display.

Reference: BB 2.2.1.

See also: 1.4-9.

-6 • Consistent Method for Data Change -6

In keyed data entry, always allow the user to change previous entries if necessary (including displayed default values) by delete and insert actions; if data change is sometimes made by direct character substitution ("typeover"), then that option should also be consistently available.

Example: Form filling may require typeover to replace displayed characters acting as field delimiters.

Example: Text editing on an electronic display can be handled with or without typeover; there seems to be no published research on the relative efficiency of user performance under these two conditions.

Comment: Using typeover, there is some risk of user confusion in replacement of an old value with a new one, during the transitional period when the item being changed is seen as a composite beginning with the new value and ending with the old. Some designers do not permit overtyping for that reason.

Comment: In some applications it may help the user to key a new entry directly above or below display of the prior entry it will replace, if that is done consistently. Here the user can compare values before confirming entry of the new data and deletion of the old.

Reference: BB 2.10; Keister and Gallaway, 1983.

-7 • User-Paced Data Entry -7

Allow users to pace their data entry, rather than having the pace being controlled by computer processing or external events.

Comment: The timing of user-paced data entry will fluctuate depending upon a user's momentary needs, attention span and time available. At maximum speed, user-paced performance is more accurate than that achieved by machine pacing.

Comment: When user pacing does not seem feasible, as in some real-time process control applications, reconsider the general approach to task allocation and interface design.

Reference: MS 5.15.2.1.1; Bertelson, Boons and Renkin, 1965.

-8 • Explicit ENTER Action -8

Always require an explicit ENTER action to initiate processing of entered data; do not initiate processing as a side effect of some other action.

Example: Returning to a menu of control options should not by itself result in computer processing of data just keyed onto a display.

Exception: In routine, repetitive data entry transactions, successful completion of one entry may automatically lead to initiation of the next, as in keying ZIP codes at an automated post office.

Comment: This practice permits the user to review data and correct errors before computer processing, particularly helpful when data entry is complex and/or difficult to reverse.

Reference: MS 5.15.2.1.4.

See also: 1.4-1, 1.4-2, 3.0-5, 4.0-2, 6.0-3, 6.3-8.

-9 o ENTER Key Labeling -9

Label an ENTER key explicitly to indicate its function.

Example: The ENTER key should not be labeled in terms of mechanism, such as CR or RETURN or XMIT.

Comment: For a novice computer user, the label should perhaps be even more explicit, such as ENTER DATA. Ideally, one consistent ENTER label would be adopted for all systems and so become familiar to all users.

Comment: Some other label might serve as well, if it were used consistently. In some current systems the ENTER key is labeled GO or DO, implying a generalized command to the computer, "Go off and do it."

Reference: PR 3.3.9.

See also: 3.0-13, 4.0-10.

-10 • Explicit CANCEL Action -10

In order to cancel a data entry, a user should always take an explicit action; data cancelation should not be accomplished as a side effect of some other action.

Example: Casual interruptions of a data entry sequence, such as paging through forms, or detouring to HELP displays, should not have the effect of erasing partially completed data entries.

Comment: If a requested sequence control action implies a more definite interruption, such as a log-off command, or a command to return to a menu display, then the user should be asked to confirm that action and alerted to the loss of any data entries that would result.

See also: Section 3.3.

-11 • Feedback for Completion of Data Entry -11

The computer should acknowledge completion of a data entry transaction with a confirmation message, if data entry was successful, or else with an error message.

Exception: In a sequence of routine, repetitive data entry transactions, successful completion of one entry might result simply in regeneration of the initial (empty) data entry display, in order to speed the next entry in the sequence.

Comment: Successful data entry should not be signaled merely by automatic erasure of entered data from the display, except possibly in the case of repetitive data entries. For single data entry transactions, it may be better to leave entered data on the display until the user takes an explicit action to clear the display.

Reference: MS 5.15.5.4.

See also: 1.0-2, 3.0-11, 4.2-1.

-12 o Feedback for Repetitive Data Entries -12

For a repetitive data entry task that is accomplished as a continuing series of transactions, indicate successful entry by regenerating the data entry display, automatically removing the just-entered data in preparation for the next entry.

Comment: Automatic erasure of entered data represents an exception to the general principle of control by explicit user action. The interface designer may adopt this approach, in the interests of efficiency, for data entry transactions that task analysis has shown will be performed repetitively.

Comment: In addition to erasure of entered data, a message confirming successful data entry might be displayed. Such a message may reassure uncertain users, especially in system applications where computer performance is unreliable.

Reference: EG 4.2.10.

See also: 1.0-2, 3.0-11, 4.2-1.

-13 o Feedback when Changing Data -13

If a user requests change (or deletion) of a data item that is not currently being displayed, then the computer should offer the option of displaying the old value before confirming the change.

Exception: Expert users may wish to implement some data changes without displayed feedback, as in so-called "global replace" transactions.

Comment: Displayed feedback will help prevent inadvertent data change, and is particularly useful in protecting delete actions. Like other recommendations intended to reduce error, it assumes that accuracy of data entry is worth extra keying by the user. For some tasks, that may not be true.

See also: 6.3-19, 6.5-10.

-14 • Keeping Data Items Short -14

For coded data items, numbers, etc., the length of an individual data entry should not exceed 5-7 characters.

Example: Coded data may include such items as badge numbers, payroll numbers, mail stops, equipment and part numbers, etc.

Comment: For coded data, lengthy items may exceed a user's memory span, inducing errors in both data entry and data review. The nine-digit ZIP codes proposed by the US Postal Service will prove difficult if not impossible to remember.

Comment: Proper names, meaningful words, and other textual material, are not coded data. Such items can be remembered more easily, and the length restriction recommended here need not apply.

Reference: BB 1.5.2; EG 6.3.3.

See also: 1.0-15.

-15 o Partitioning Long Data Items -15

When a long data item must be entered, it should be partitioned into shorter symbol groups for both entry and display.

Example: A 10-digit telephone number can be entered as three groups, NNN-NNN-NNNN.

Reference: BB 1.4.1; MS 5.15.3.1.7, 5.15.3.5.7, 5.15.3.5.8.

See also: 1.0-14, 2.1.2-14.

-16 • Optional Abbreviation -16

Permit optional abbreviation of lengthy data items to minimize data entry keying by expert users, when that can be done without ambiguity.

Comment: Novice and/or occasional users may prefer to make full-form entries, while experienced users will learn and benefit from appropriate abbreviations.

Reference: BB 2.4.1; EG 6.3.5; MS 5.15.2.2.7.

-17 o Distinctive Abbreviation -17

When using abbreviations or other codes to shorten data entry, design them to be distinctive in order to avoid potentially confusing similarity.

Example: BOS vs. LAS is good; but LAX vs. LAS risks confusion.

Reference: BB 3.1; MS 5.15.2.1.10.

-18 o Consistent Abbreviation Rule -18

When defining abbreviations, follow some consistent abbreviation rule that can be explained to users.

Example: Simple truncation is probably the best choice, when that can be done without ambiguity.

Comment: When encoding abbreviations for data entry the user must know what the rule is. Truncation provides inexperienced users with a straightforward and highly successful method for generating abbreviations, and is a rule that can be easily explained. Moreover, truncation works at least as well, and sometimes better than, more complicated rules, such as word contraction with omission of vowels.

Comment: Designers of military systems may wish to consult the relevant standard for abbreviations, MIL-STD-12D.

Reference: Ehrenreich and Porcu, 1982; Hirsh-Pasek, Nudelman and Schneider, 1982; Moses and Ehrenreich, 1981.

-19 o Minimal Exceptions to Abbreviation Rule -19

Use special abbreviations (i.e., those not formed by consistent rule) only when they are required for clarity.

Comment: Special abbreviations will be needed to distinguish between words whose abbreviations by rule are identical, or when abbreviation by rule forms another word, or when the special abbreviation is already familiar to system users. If more than 10 percent of abbreviations are special cases, consider changing the abbreviation rule.

Reference: Moses and Ehrenreich, 1981.

-20 o Minimal Deviation from Abbreviation Rule -20

When an abbreviation must deviate from the consistent rule, minimize the extent of deviation.

Example: In abbreviation by truncation, letters in the truncated form should be changed one at a time until a unique abbreviation is achieved.

Reference: Moses and Ehrenreich, 1981.

-21 o Fixed Abbreviation Length -21

Make abbreviations the same length, the shortest possible that will ensure unique abbreviations.

Comment: Desirable length will depend upon the vocabulary size of words to be abbreviated. For a vocabulary of 75 words, 4-letter abbreviations might suffice. For smaller vocabularies, still shorter abbreviations might be used.

Reference: Moses and Ehrenreich, 1981.

-22 o Clarifying Unrecognized Abbreviations -22

When abbreviated data entries are not recognized, the computer should apply data validation routines and interrogate the user as necessary to resolve any ambiguity.

Example: This may occur when a user enters a misremembered abbreviation.

See also: 6.0-7.

-23 • Prompting Data Entry -23

The computer should prompt users about required format and acceptable values for data entries.

Example: (Good) Vehicle type: __
c = Car
t = Truck
b = Bus

(Bad) Vehicle type: __

Exception: Prompting is needed especially by novice or infrequent users. Prompting may not be needed by skilled users, however, and indeed may hinder rather than help performance in situations where display output is slow (as with teletype displays).

Comment: Prompting is particularly needed for coded data entries. Menu selection may be appropriate for this purpose, since that dialogue mode does not require the user to remember codes but merely to choose among displayed alternatives. Other methods of prompting include labeling data fields (e.g., VEHICLE TYPE C/T/B), and/or providing optional guidance displays.

Reference: Gade, Fields, Maisano, Marshall, and Alderman, 1981; Seibel, 1972.

See also: 1.4-5, Section 3.1.3, 4.4-7.

-24 • Character Entry via Single Keystroke -24

Allow users to enter each character of a data item with a single stroke of an appropriately labeled key.

Example: When a keyboard is intended primarily for numeric input, with several letters grouped on each key such as a telephone keypad, do not require a user to make alphabetic entries by double keying.

Comment: Devices that involve complex keying methods for alphabetic entry (e.g., pressing more than one key, simultaneously or successively) require special user training and risk frequent data entry errors.

Comment: When hardware limitations seem to require double keying of alphabetic entries, try to limit data codes so that only (single-keyed) numeric entries are required. Alternatively, consider providing software to interrogate the user to resolve any ambiguities resulting from single keying of alphabetic entries.

Reference: Butterbaugh and Rockwell, 1982; Smith and Goodwin, 1971a.

-25 o Minimal Shift Keying -25

Design data entry transactions to minimize the need for shift keying.

Comment: Shift keying can be considered a form of double keying, which imposes a demand for extra user attention. Keyboard designers should put frequently used characters where they can be easily keyed. Conversely, software designers should avoid frequent use of characters requiring shift keying.

Reference: EG 6.3.12.

-4 o Control Entries Distinct from Text -4

If control entries are made by keying onto the display, such as by keyed menu selections or commands, ensure that they will be distinguishable from displayed text.

Example: Keyed control entries might be made only in a reserved window in the display.

Comment: The intent here is to help ensure that a user will not inadvertently enter controls as text, or vice versa. If a command entry is keyed into the body of a text display, perhaps at the end of the last sentence, then a user cannot be certain whether the computer will interpret the command as a text entry or as a control entry.

Comment: In applications where the screen cannot display all possible format features (e.g., special fonts), format codes representing those features are usually displayed within the text. It is not practical in such cases to display format codes in a separate window, since a displayed code must mark the text that will be affected by the code. These codes should therefore be highlighted in some way to distinguish them from text.

Comment: One way of avoiding the problem altogether is to use function keys rather than command entry to control text editing. To provide a general range of text editing functions, however, many keys will be needed. A practical design approach might be to adopt double-keying logic for all keys on a standard (QWERTY) keyboard, where control-F means FILE a document, control-G means GET a document, etc., and providing appropriate extra labels for those keys.

-5 • Natural Units of Text -5

Allow users to specify segments of text in whatever units are natural for entry/editing.

Example: For unformatted ("free") text, natural units will be characters, words, phrases, sentences, paragraphs, and pages; for specially formatted text, such as computer program listings, allow specification of other logical units, including lines, subsections, sections, etc.

-2 • Full Editing Capabilities During Text Entry -2

Provide users a full range of editing capabilities during text entry, and do not require users to distinguish between separate entry and editing modes.

Comment: Some systems are designed to distinguish between the tasks of creating and editing a document, allowing users to make only small changes while creating a document, but permitting larger changes during subsequent editing. Such a design approach assumes that a user will first create an entire document, probably typing from a written draft, and will then edit the document as a separate task. With increasing use of on-line text processing, this distinction between creation and editing will become blurred. Users may wish to make large editing changes during initial document creation, and conversely may wish to insert large sections of new material during later editing.

See also: 2.0-8.

-3 • Free Cursor Movement -3

For text editing, allow users to move the cursor freely over a displayed page of text to specify items for change, and to make changes directly to the text.

Comment: Free cursor movement and changes made directly to the text are characteristics usually associated with so-called screen-based editors and not associated with line- or command-based editors. Screen-based editors are preferred by users and are potentially more efficient.

Reference: MS 5.15.3.8.2; Gould, 1981; Roberts and Moran, 1983; Shneiderman, 1982.

See also: 1.0-1, 2.6-8.

Text entry refers to the initial entry and subsequent editing of textual material, including messages.

-1 • Adequate Display Capacity -1

Ensure that display capacity, i.e., number of lines and line length, is adequate to support efficient performance of text entry/editing tasks.

Example: For text editing where the page format of subsequent printed output is critical, the user's terminal should be able to display full pages of text in final output form, which might require a display capacity of 50-60 lines or more.

Example: For general text editing where a user might need to make large changes in text, i.e., sometimes moving paragraphs and sections, a display capacity of at least 20 lines should be provided.

Example: Where text editing will be limited to local changes, i.e., correcting typos and minor rewording, as few as seven lines of text might be displayed.

Comment: A single line of displayed text should not be used for text editing. During text editing, a user will need to see some displayed context in order to locate and change various text entries. Displaying only a small portion of text will make a user spend more time moving forward and back in a displayed document to see other parts, will increase load on the user's memory, and will cause users to make more errors.

Reference: Darnell and Neal, 1983; Elkerton, Williges, Pittman and Roach, 1982.

See also: 1.3-28.

Direction Designation refers to user entry of directional data (azimuth, bearing, heading, etc.) on a display.

-1 • Analog Entry of Estimated Direction -1

When direction designation is based on graphic representation, provide for some "analog" means of entry, such as vector rotation on the display, and/or a suitably designed rotary switch.

Example: Heading estimation for displayed radar trails.

Exception: When approximate direction designation will suffice, for just eight cardinal points, keyed entry can be used.

Comment: In matching graphic display, an entry device providing a visual analog will prove both faster and more accurate.

Reference: Smith, 1962a.

-2 • Keyed Entry of Quantified Direction -2

When designation of direction is based on already quantified data, permit keyed entry.

-23 • Display Format Protection -23

When there are areas of a display in which data entries cannot be made (blank spaces, protected field labels, etc.), prevent the cursor from entering those areas, and make those areas insensitive to pointing actions.

Exception: When a user may have to modify display formats, then this automatic format protection can be provided as a general default option subject to user override.

Comment: Automatic format protection will generally make cursor positioning easier for a user, since the cursor will not have to be stepped through blank areas, and much routine cursor control can be accomplished with only casual reference to the display.

Reference: BB 1.8.13; EG 7.5; MS 5.15.4.3.12; PR 3.3.2.

See also: 1.4-7, 2.0-9, 6.2-5.

-24 • Data Entry Independent of Cursor Placement -24

Ensure that an ENTER action for multiple data items results in entry of all items, regardless of where the cursor is placed on the display.

Comment: A user may choose to move the cursor back to correct earlier data items, and may not move the cursor forward again. The computer should ignore cursor placement in such cases.

See also: 6.5-9.

-20 • Consistent HOME Position -20

When there is a predefined HOME position for the cursor, which is usually the case, that position should be consistent on displays of a given type.

Example: HOME might be in the upper left corner of a text display, or at the first field in a form-filling display, or at the center of a graphic display.

Comment: The HOME position of the cursor should also be consistent in different windows/sections of a partitioned display.

Reference: MS 5.15.2.1.8.3.

See also: 4.4-12.

-21 • Consistent Cursor Placement -21

On initial appearance of a data entry display, the cursor should appear automatically at some consistent and useful location.

Example: In a form-filling display, the cursor should be placed in the first entry field.

Reference: BB 2.1.4; MS 5.15.4.3.6.

See also: 1.4-26, 4.4-12.

-22 • Easy Cursor Movement to Data Fields -22

If a cursor must be positioned sequentially in predefined areas, such as displayed data entry fields, ensure that this can be accomplished by simple user action.

Example: Programmable tab keys are customarily used for this purpose.

Comment: Automatic cursor advance is generally not desirable.

Reference: MS 5.15.4.3.6.

See also: 1.4-24.

-16 • Minimal Use of Multiple Cursors -16

Employ multiple cursors on a single display only when they are justified by careful task analysis.

Example: Multiple cursors might be useful to mark a user's place when manipulating data in multiple display windows.

Example: In graphic interaction, one cursor might be used for line drawing and a different cursor for alphanumeric data entry (labels, etc.).

Comment: Multiple cursors may confuse a user, and so require special consideration if advocated in USI design.

-17 o Distinctive Multiple Cursors -17

If multiple cursors are used, make them visually distinctive from one another.

See also: 1.1-1.

-18 o Distinctive Control of Multiple Cursors -18

If multiple cursors are controlled by a single device, provide a clear signal to the user to indicate which cursor is currently under control.

-19 o Compatible Control of Multiple Cursors -19

If multiple cursors are controlled by different devices, ensure that their separate controls are compatible in operation.

Example: Assume that one cursor is moved upward on a display by forward motion of a joystick. Then a second cursor should also be moved upward by forward motion, perhaps by forward motion of a second joystick, or by forward motion of a thumbwheel or some other such device.

Reference: Morrill and Davies, 1961.

See also: 1.1-15, 3.0-13.

-13 o Large Pointing Area for Option Selection -13

In selection of displayed alternatives, design the acceptable area for pointing (i.e., cursor placement) to be as large as consistently possible, including at least the area of the displayed label plus a half-character distance around the label.

Comment: The larger the effective target area, the easier the pointing action will be, and the less risk of error in selecting the wrong label by mistake. Some researchers have recommended a target separation on the display of no less than 6 mm.

Reference: BB 2.12; EG 2.3.13, 6.1.3; Whitfield, Ball and Bird, 1983.

See also: 3.1.3-5.

-14 • Cursor Control at Keyboard -14

When position designation is required in a task emphasizing keyed data entry, permit cursor control by some device integral to the keyboard (function keys, joystick, "cat", etc.).

Comment: Separately manipulated devices (lightpen, "mouse", etc.) will tend to slow the user.

Reference: Foley and Wallace, 1974.

See also: 1.0-4.

-15 • Compatible Control of Cursor Movement -15

Ensure that control actions for cursor positioning are compatible with movements of the displayed cursor, in terms of control function and labeling.

Example: For cursor control by key action, a key labeled with a left-pointing arrow should move the cursor leftward on the display; for cursor control by joystick, leftward movement of the control (or leftward pressure) should result in leftward movement of the cursor; etc.

See also: 1.1-19, 3.0-13.

-10 o Proportional Spacing -10

If proportional spacing is used for displayed text, the computer should make necessary adjustments automatically when the cursor is being positioned for data entry or data change.

Example: Automatic proportional spacing is useful for cursor control when editing text composed for type setting.

Exception: Manual override may help a user in special cases where automatic spacing is not wanted.

Comment: Without automatic computer aids, a user probably will not handle proportional spacing accurately.

-11 • Continuous Cursor Positioning -11

For continuous position designation, such as needed for line drawing, provide a continuously operable control (e.g., joystick) rather than requiring a user to take incremental, discrete key actions.

-12 • Direct Pointing -12

When position designation is the sole or primary means of data entry, as in selection among displayed alternatives, permit cursor placement by direct pointing (e.g., with a touch display or lightpen) rather than incremental stepping or slewing controls (e.g., keys, joystick, etc.).

Reference: MS 5.15.2.5.1; Albert, 1982; Goodwin, 1975.

-6 • Stable Cursor -6

The displayed cursor should be stable, i.e., it should remain where it is placed until moved by the user (or computer) to another position.

Comment: Some special applications, such as aided tracking, may benefit from computer-controlled cursor movement. The intent of the recommendation here is to avoid unwanted "drift".

Reference: EG 6.1.

-7 • Responsive Cursor Control -7

For arbitrary position designation, moving a cursor from one position to another, the cursor control should permit both fast movement and accurate placement.

Comment: Rough positioning should take no more than 0.5 seconds for full screen traversal. Fine positioning may require incremental stepping of the cursor, or a control device incorporating a large control/display ratio for small displacements, or a selectable vernier mode of control use. For any given cursor control action, the rate of cursor movement should be constant, i.e., should not change with time.

-8 • Consistent Incremental Positioning -8

When cursor positioning is incremental by discrete steps, the step size of cursor movement should be consistent horizontally (i.e., in both right and left directions), and consistent vertically (in both up and down directions).

Comment: Horizontal and vertical step sizes need not be the same, and in general will not be.

-9 o Variable Step Size -9

When character size is variable, incremental cursor positioning should vary correspondingly, with a step size matching the size of currently selected characters.

-3 o Precise Pointing -3

When fine accuracy of positioning is required, as in some forms of graphic interaction, design the displayed cursor to include a point designation feature.

Example: A cross may suffice (like cross-hairs in a telescope), or perhaps a notched or V-shaped symbol (like a gun sight).

Comment: Precise pointing will also require a cursor control device capable of precise manipulation. Touch displays, for example, will not permit precise pointing.

Reference: MS 5.15.2.1.8.2; Whitfield, Ball and Bird, 1983.

-4 • Explicit Activation -4

Require users to take a separate, explicit action, distinct from cursor positioning, for the actual entry (enabling, activation) of a designated position.

Exception: For line drawing or tracking tasks the need for rapid, continuous entry may override the need to reduce entry errors.

Reference: MS 5.15.2.5.4; Albert, 1982; Foley and Wallace, 1974; Whitfield, Ball and Bird, 1983.

See also: 3.1.3-6, 6.0-3.

-5 • Fast Response -5

The computer should acknowledge entry of a designated position within 0.2 seconds.

Example: Almost any consistently provided display change will suffice to acknowledge pointing actions, such as brightening or flashing a selected character. In some applications it may be desirable to provide a more explicit message indicating that a selection has been made.

Reference: EG Table 2; MS 5.15.8.

See also: 1.0-2, 4.2-2, 4.2-10.

Position Designation refers to user selection and entry of a position on a display, or of a displayed item.

-1 • Distinctive Cursor -1

For position designation on an electronic display, provide a movable cursor with distinctive visual features (shape, blink, etc.).

Exception: When position designation involves only selection among displayed alternatives, highlighting selected items might be used instead of a separately displayed cursor.

Comment: When choosing a cursor shape, consider the general content of the display. For instance, an underscore (__) cursor would be difficult to see on a display of underscored text, or on a graphical display containing many other lines.

Comment: If the cursor is changed to denote different functions (e.g., to signal deletion rather than entry), then each different cursor should be distinguishable from the others.

Comment: If multiple cursors are used on the same display (e.g., one for alphanumeric entry and one for line drawing), then each cursor should be distinguishable from the others.

Reference: Whitfield, Ball and Bird, 1983.

See also: 1.1-17, 4.0-9.

-2 o Non-Obscuring Cursor -2

Design the cursor so that it does not obscure any other character displayed in the position designated by the cursor.

Example: A block cursor might employ brightness inversion ("reverse video") to show any other character that it may be marking.

-29 • Single/Multiple Blanks Equivalent -29

The computer should treat single and multiple blank characters as equivalent in data entry; do not require users to count blanks.

Comment: People cannot be relied upon to pay careful attention to such details. The computer should handle them automatically, e.g., ensuring that two spaces follow every period in text entry (if that is the desired convention), and spacing other data items in accord with whatever format has been defined.

See also: 3.1.5-15.

-26 • Upper/Lower Case Equivalent -26

For coded data entry, treat upper and lower case letters as equivalent.

Comment: For data codes, users find it difficult to remember whether upper or lower case letters are required, and so the software design should not try to make such a distinction. For text entry, however, conventional use of capitalized letters should be maintained.

See also: 1.3-10.

-27 • Decimal Point Optional -27

Entry or omission of a decimal point at the end of a number should be optional.

Example: An entry of "56." should be processed as equivalent to an entry of "56", and vice versa.

Comment: If a decimal point is required for data processing, the computer should probably be programmed to append one as needed. Most users will forget to do it.

Reference: Keister and Gallaway, 1983.

-28 • Leading Zeros Optional -28

Entry of leading zeros should be optional for general numeric data.

Example: If a user enters "56" in a field that is four characters long, the system should recognize the entry rather than requiring that "0056" be entered.

Exception: Special cases such as entry of serial numbers or other numeric identifiers.

Reference: BB 2.2.3; EG 6.3.11.

-6 o Control Entry Based on Units of Text -6

Allow users to specify units of text as modifiers for control entries.

Example: Consider two alternative control sequences to delete a four character word:

(Good) DELETE WORD

(Bad) DELETE DELETE DELETE DELETE

Comment: Control entries, whether accomplished by function key, menu selection, or command entry, will be easier and more powerful when a user can specify text in natural units, rather than having to repeat an entry for each text character.

Comment: When units of text are modifiers for all control entries, the syntax for those control entries will be easier to learn. Whether a control action is to MOVE or to DELETE, the modifiers to specify text are the same.

Reference: MS 5.15.3.8.4.1, 5.15.3.8.4.2.

See also: 3.0-6, 4.0-1.

-7 o Highlighting pecified Text -7

When text has been specified to become the subject of control entries, highlight that segment of text in some way to indicate its boundaries.

Comment: Text may be specified for various purposes -- for underlining or bolding, moving, copying, or deleting. Highlighting provides the user with direct feedback on the extent and content of specified text, reducing the likelihood of specification errors.

See also: 4.2-10.

-8 o Cursor Movement by Units of Text -8

Allow users to move the cursor by specific units of text, as well as one character at a time.

Comment: The time necessary to position a cursor is directly related to the number of control actions required. Incremental cursor movement by character will therefore be inefficient when mo ing the cursor over large units of text.

Comment: Cursor positioning will be easier if appropriate function keys can be provided. A SENTENCE key that allows a user to move directly to the next displayed sentence will be more convenient than some double-keying logic such as CONTROL-S.

See also: 1.1-12.

-9 • String Search -9

Allow users to specify a string of text and request the computer to advance (or back up) the cursor automatically to the next (or last previous) occurrence of that string.

Comment: Novice users may prefer to move through a displayed document by units of text, such as by word or paragraph. More experienced users, however, may sometimes wish to specify cursor placement directly. An automatic string search capability will generally speed cursor placement in comparison with incremental positioning, particularly when moving over large portions of a document.

Reference: Elkerton, Williges, Pittman, and Roach, 1982.

-10 o Upper/Lower Case Equivalent in Search -10

Unless otherwise specified by a user, the computer should treat upper and lower case letters as equivalent in searching text.

Example: "STRING", "String", and "string" should all be recognized/accepted by the computer when searching for that word.

Comment: In searching for words, users will generally be indifferent to any distinction between upper and lower case. The computer should not compel a distinction that users do not care about and may find difficult to make. In situations when case actually is important, allow users to specify case as a selectable option in string search.

Comment: It may also be useful for the computer to ignore such other features as bolding, underlining, parentheses and quotes when searching text.

See also: 1.0-26.

-11 o Specifying Case in Search -11

When case is important, allow users to specify case as a selectable option in string search.

Example: When searching a document in which all the headings are capitalized, a user might wish to find a string only when it appears in a heading.

Comment: Users may also wish to specify features such as bolding, underlining, and quotes when searching text.

-12 o Global Search and Replace -12

When systematic editing changes will be made throughout a long document, consider providing a "global search and replace" capability in which the computer will replace all occurrences of one text string with another without requiring the user to confirm each change.

Comment: Using such a capability will always entail some risk of unwanted change.

-13 o Case in Global Search and Replace -13

If a global search and replace capability is provided, each time the string is replaced the computer should change the case of the new string to match the case of the old string, unless otherwise specified by a user.

Example: If a word is replacing the first word in a sentence, the first letter of the new word should be capitalized; if it is replacing a word that is entirely in lower case, the new word should also be in lower case.

Comment: On occasion, however, a user might wish to replace an erroneous lower-case word ("Mitre") with a correctly capitalized version ("MITRE").

-14 • Automatic Pagination Aids -14

Provide automatic pagination for text entry/editing, allowing users to specify the page size.

Exception: For short documents, automatic pagination may not be needed.

-15 o User Control of Pagination -15

When automatic pagination is provided, allow users to override the pagination and specify that a new page be started at any point in the document.

-16 o Controlling Integrity of Text Units -16

When automatic pagination is provided, allow users to specify the number of lines in a paragraph that will be allowed to stand alone at the top or bottom of a page (i.e., the size of "widows" and "orphans"), and to specify any text that should not be divided between two pages, such as inserted lists or tables.

-17 • Automatic Line Break -17

For entry/editing of unformatted text, provide an automatic line break ("carriage return") when text reaches the right margin, with provision for user override.

Comment: For specially formatted text, such as computer program listings, users may need to control line structure themselves and hence need to override any automatic line break. Even when entering unformatted text, a user will sometimes wish to specify a new line at some particular point, if only for esthetic reasons.

-18 o Consistent Word Spacing -18

Unless otherwise specified by the user, entered text should be left-justified to maintain constant spacing between words, leaving right margins ragged if that is the result.

See also: 2.1.1-5.

-19 o Hyphenation by Users -19

In the entry/editing of text, automatic pagination and line breaks by the computer should keep words intact, and introduce hyphenation only where specified by users.

Comment: Where compound words have been hyphenated by a user, the computer might break the compound after a hyphen, for pagination or line breaks, unless otherwise specified by the user. Compound words formed with slashes (e.g., "entry/editing") might be treated in a similar manner.

See also: 2.1.1-6.

-20 • Format Control by User -20

Provide easy means for users to specify required format control features during text entry/editing, e.g., to specify margin and tab settings.

Example: One convenient method of margin and tab control is to allow users to mark settings on a displayed "ruler" that extends the width of a page and is continuously displayed at the top of the screen.

Comment: Required format features will vary depending on the application. For instance, font size may be quite important when composing text for typsetting but unnecessary when editing computer programs. The intent of this guideline is that all required format features should be easy to control, and should take priority in interface design. Any format features which are provided but are optional for the user's task should not be made easy to use at the expense of required format features.

-21 • Establishing Predefined Formats -21

When text formats must follow predefined standards, the computer should provide the standard format automatically; do not rely on users to remember and specify proper formats.

Example: Standard formats might be required for letters, memos, or other transmitted messages.

See also: 5.1-2.

-22 o Storing User-Defined Formats -22

When text formats cannot be predicted in advance, which is often the case, allow users to specify and store for future use the formats that might be needed for particular applications.

Example: A special format might be adopted for generating a particular report at periodic intervals.

-23 • Moving Text -23

Allow users to select and move text segments from one place to another within a document.

Comment: A user should not have to re-enter (i.e., rekey) text that is already available to the computer.

Comment: One convenient method of allowing the user to both move and copy text is to provide a "cut and paste" facility in which the "cut" text remains in a storage buffer and can be "pasted" more than once. For copying, the user can cut text, paste it back into its original location, and paste it again at a new location.

-24 • Storing Frequently Used Text -24

Allow users to label and store frequently used text segments, and later to recall (copy into current text) stored segments identified by their assigned labels.

Example: Much text processing involves repetitive elements, such as signature blocks, technical terms, long names, formulas or equations, specific to different applications.

-25 • Necessary Data Displayed -25

Ensure that whatever information a user needs for text entry/editing is available for display, as an annotation to displayed text.

Example: A user might wish to see format control characters, such as tab and margin settings.

Comment: Required annotation will vary with the application. Some annotation may be so commonly needed that it should be continuously displayed -- e.g., document name, page number, indication of control mode (if any), etc. Other annotation might be displayed only at user request -- such as document status (date last changed, last printed, etc.), which might be displayed in an optional window overlay, and format control characters, which might be visible as an optional text display mode.

-26 o Text Distinct from Annotation -26

Ensure that annotations to displayed text are distinguishable from the text itself.

Example: Continuous annotation might be displayed in the top and/or bottom lines of a page, separated from the text by blank lines; optional annotation might be displayed in window overlays; format control characters might be shown in a special display mode where text has been expanded to permit annotation between lines.

-27 • Printing for Proofreading -27

If proofreading will be an important part of text entry/editing, provide convenient printing facilities so that users can read printed drafts rather than proofread from an electronic display.

Comment: Proofreading from electronic displays is slower and less accurate than reading printed text. Not all users will wish to proofread from printed copy, but that option should be available for those who do.

Comment: A user who must wait hours to obtain a printout will be unlikely to do so just to proofread a draft. To encourage proofreading from printed copy, the printer must be conveniently located and provide quick output.

Reference: Wright and Lickorish, 1983.

See also: 2.5-9.

-28 • Text Displayed as Printed -28

Allow users to display text exactly as it will be printed.

Comment: Accurate display is particularly necessary when the format of printed output is important, as when printing letters, tables, etc.

Comment: Ideally, text displays should be able to represent all the features that are provided in printed output, including upper and lower case, underlining, bolding, subscripting, superscripting, special symbols, and different styles and sizes of type. When those features are important, the necessary display capability should be provided.

Comment: For special formatting features that are not frequently used, it may be sufficient to use extra symbols to note text features that cannot be directly displayed. In that case, care should be taken that such annotation does not disturb the spacing of displayed text. This may require two display modes, one to show text spacing as it will be printed and the other to show annotations to the text.

Comment: A corollary to this recommendation is that changes made to displayed text should appear as a user makes them. Some line-based editors show changes only after a document has been filed and later recalled for display, which does not represent good user interface design.

Reference: Foley and Van Dam, 1982 (p. 15); Gould, 1981.

See also: 1.3-1, 1.3-26.

-29 • Flexible Printing Options -29

In printing text, allow users to select among available output formats (line spacing, margin size, etc.) and to specify the parts of a document to be printed; do not require that an entire document be printed.

Example: Permit a user to print just those portions of a document that have been changed, perhaps specifying just the first page, or page 17, or the last five pages, etc.

Comment: This is particularly important when long documents will be edited. A user should not be required to print an entire 50-page document just because of a change to one page.

-30 • Information on Printing Status -30

Inform users concerning the status of requests for printouts.

Example: The computer should acknowledge print requests immediately, and might provide a subsequent message to indicate when a printout has been completed if the printer is remote (unobservable) from the user's work station.

Example: If there is a queue of documents waiting for printout, a user should be able to get an estimate as to when a particular document will be printed.

Comment: If a user is responsible for operating a local printer, the computer might display messages to alert the user of potential malfunctions, e.g., if its paper supply is exhausted, if the paper is not correctly loaded, etc.

See also: 3.0-11, 4.2-5.

-31 • Auditory Signals for Alerting Users -31

During text entry/editing, provide an auditory signal whenever it is necessary to draw a user's attention to the display.

Comment: A touch typist entering text from written copy will often not be looking at the display screen, and therefore may not notice visual indicators of errors or mode changes unless they are accompanied by auditory signals.

See also: 2.4-35.

-32 • Protecting Text During Page Overruns -32

When a user is inserting text into a document that has already been paginated, ensure that no text is lost if the user inserts more text than a page can hold.

Comment: It is difficult for a user to keep track of page size, particularly if the size of the display screen is less than the full page specified for printed text, which is often the case. A user will often not know when more text has been inserted into a page than there is room for. The computer should accommodate text insertions with automatic repagination.

-33 • Confirming Actions in DELETE Mode -33

If a DELETE mode is used, highlight any text specified by a user for deletion and require the user to confirm the DELETE action.

Comment: Requiring a user to confirm actions in DELETE mode is particularly important when the control entries for cursor positioning (e.g., WORD, SENTENCE, PARAGRAPH, PAGE) are also used to specify text for deletion, which is often the case. Users will associate the specification of text units primarily with cursor positioning, which is the most frequent action in text editing. In a DELETE mode, after specifying text units for deletion, a user may press a PARAGRAPH key intending to move to the next paragraph but accidentally delete the next paragraph. Confirmation of DELETE actions will tend to prevent such errors.

Comment: An alternative approach to this problem is not to provide a continuing DELETE mode, but instead require double keying to accomplish deletions. In a DELETE mode, a user might press a DELETE key followed by unlimited repetitions of a WORD key (or keys specifying other units of text). With double keying, the user would have to press DELETE before each selection of a text unit to be deleted.

See also: 1.3-6, 6.3-20, 6.5-18.

-34 • Reversible Actions -34

Any user action should be reversible.

Example: If a user centers a heading and then decides it would look better flush against the left margin, an UNDO action should reverse the centering and move the heading back to its original location.

Example: If a user underlines a paragraph of text and then decides it should be in all capital letters instead, an UNDO action should reverse the underlining. The user should not be required to delete the paragraph and retype it just to erase the underscoring.

Comment: Reversible actions are particularly important in a text editing environment because text formatting often involves experimentation with features such as underscoring, bolding, and spacing. If users know that they can reverse whatever they do, they will feel more free to delete text and experiment with formatting features.

Comment: An UNDO capability is currently available in some interface designs. In some applications, however, this capability is provided through the use of an UNDO key which can only reverse the most recent control action. For text editing, users must be able to reverse such formatting features as centering and bolding at any time. Therefore, if control actions are to be made reversible, an UNDO action should be able to reverse more than the most recent command, perhaps by requiring the user to specify which command to undo, and/or to place the cursor at the location of the format feature that is to be reversed.

Comment: When text segments have been deleted, it should be possible to retrieve more than the most recent deletion. Some systems do this by storing all deletions in a special file. Deleted text which the user wishes to retrieve can then be moved from the deletion file to the file in which the user is presently working.

Reference: Lee and Lochovsky, 1983; Nicherson and Pew, 1971; Shneiderman, 1982.

See also: 3.5-10, 6.5-20.

-35 • User Confirmation of Editing Changes -35

When a user signals completion of document editing, allow the user to confirm that changes should be made to the original document, or else to choose alternative options.

Comment: A user will generally wish to replace the original document with its edited version. However, sometimes a user may decide that editing mistakes have been made, and wish to discard the new version while saving the original. Or a user might wish to save the new version as a separate document, while saving the original unchanged. Such decisions can be made best at the end of an editing session, when the user knows what has been accomplished, rather than before a session is begun.

Comment: During text editing, the computer should always retain a copy of the original document until the user confirms that it should be changed. The original document should not be changed automatically as the user enters each editing change.

See also: 6.3-20, 6.5-18.

(Bad) Sample Data Form (Bad)

```
Name Andrew D. Jones                    Visa Number 356478

Birthplace London                       Nationality English

Passport Z196284                        Birthdate Mar. 22,

Address  1925
         5 Fairview Lane, Loughborough, L
         E11 3RG, England

Other travelers on this visa
Traveler's Name                    Date of Birth - Place
Sandra J. Jones                         Oct. 11, - 1928
Birmingham
Cynthia L. Jones                        June 12, - 1968
Paris, France

Press ENTER when done
```

This bad data form display violates in some degree several design guidelines in this section:

- 1.4- 3 Minimal use of delimiters
- 6 Consistent labeling
- 9 Marking field boundaries
- 10 Prompting field length
- 13 Explicit tabbing to data fields
- 14 Distinctive label format
- 16 Label punctuation as entry cue
- 17 Informative labels
- 18 Data format cueing in labels
- 23 Form compatible with source documents

This bad data form also violates various design guidelines pertaining to data display, as noted at the end of Section 2.1.2.

(Good) Sample Data Form (Good)

```
                    VISA APPLICATION

NAME: Jones, Andrew David_______          VISA: 356 478
      LAST, FIRST MIDDLE

BIRTH COUNTRY: UK     DATE:  3/22/25
                             MM DD YY

  NATIONALITY: UK     PASSPORT: Z196284__

      ADDRESS: 5 Fairview Lane_____________
               Loughborough, LE11 3RG_______
               England______________________

OTHER TRAVELERS ON THIS VISA
                                  BIRTH
NAME:                             COUNTRY:   DATE:
Jones, Sandra Jean_____           UK         10/11/28
Jones, Cynthia Leigh____          FR          6/12/68
_______________________           __         __/__/__
_______________________           __         __/__/__
LAST, FIRST MIDDLE                           MM DD YY

* Press ENTER when done.
```

These sample displays represent a possible form for entry and review of visa application data. In the good form, data entries are bolded to help distinguish them from labels and field delimiters. Fields are ordered consistently in relation to a (supposed) paper application form, and formatted to facilitate both data entry and data review.

The bad display is annotated to indicate violations of several of the design guidelines proposed here for data forms. The data entries in the bad display have been invented to suggest what a user might produce if confused by inadequate labeling and the absence of field delimiters.

-25 o Data Items in Logical Order -25

If no source document or external information is involved, then design forms so that data items are ordered in the sequence in which a user will think of them.

Comment: The software designer will need to work with prospective system users to determine what represents a logical sequence of data entries.

Reference: PR 4.8.5.

See also: 2.3-13.

-26 • Automatic Cursor Placement -26

When a form for data entry is displayed, the computer should place the cursor automatically at the beginning of the first entry field.

Exception: If a data form is regenerated following an entry error, the cursor should be placed in the first field in which an error has been detected.

Reference: BB 2.1.4; PR 4.9.1.

See also: 1.1-21, 2.1.3-27, 4.4-12.

-23 o Form Compatible with Source Documents -23

When data entry involves transcription from source documents, ensure that form-filling displays match (or are compatible with) those documents, in terms of item ordering, data grouping, etc.

Example: [See sample displays at the end of this section.]

Comment: If paper forms are not optimal for data entry, consider revising the layout of the paper form.

Comment: If data entries must follow an arbitrary sequence of external information (e.g., keying telephoned reservation data), employ some form of command language dialogue instead of form filling, to identify each item as it is entered so that the user does not have to remember and re-order items.

Reference: BB 1.8.9; MS 5.15.3.1.1.b, 5.15.4.3.3; PR 4.8.3, 4.8.5, 4.10.7; Stewart, 1980.

See also: 2.3-1, 2.3-13, 3.1.1-4, 4.0-6.

-24 • Minimal Cursor Positioning -24

When designing displays for form-filling data entry, minimize user actions required for cursor movement from one field to the next.

Comment: Placing all required fields before any optional fields will sometimes make data entry more efficient.

Reference: BB 2.1.3.

See also: 1.1-22.

-21 o Alternative Units of Measurement -21

When alternative measurement units are acceptable, provide space in the data field for user entry of a unit designator.

Example: DISTANCE: _ _ _ _ (MI/KM) _ _

Reference: MS 5.15.4.3.10; PR 4.8.11.

See also: 4.0-11.

-22 • Form Compatible for Data Entry and Display -22

When forms are used for reviewing displayed data as well as for data entry, make the form for data entry compatible with that for display output; use the same item labels and ordering for both.

Comment: When a display format optimized for data entry seems unsuited for data display, or vice versa, some compromise format should be designed, taking into account the relative functional importance of data entry and data review in the user's task.

Reference: MS 5.15.3.1.1.a.

See also: 2.1.2-12, 2.3-1, 4.0-6.

-18 • Data Format Cueing in Labels -18

Labels for data fields may incorporate additional cueing of data formats when that seems helpful.

Example: DATE (MM/DD/YY): __ __/__ __/__ __

Example: [See sample displays at the end of this section.]

Reference: BB 2.1.8; MS 5.15.4.3.5; PR 4.8.9.

See also: 4.0-14.

-19 o Labeling Units of Measurement -19

When a measurement unit is consistently associated with a particular data field, include that unit as part of the field label, rather than requiring a user to enter it.

Example: COST: $ __ __ __ __

Example: SPEED (MPH): __ __ __

Reference: BB 2.2.6; MS 5.15.4.3.10; PR 4.8.11.

See also: 2.1.2-10, 4.0-11.

-20 o Familiar Units of Measurement -20

Employ units of measurement that are familiar to the user.

Example: (Good) SPEED LIMIT: __ __ miles per hour

FUEL USE: __ __.__ miles per gallon

(Bad) SPEED LIMIT: __ __ feet per second

FUEL USE: .__ __ gallons per minute

Comment: If data must be converted, program the computer to handle that automatically.

Reference: BB 2.3; MS 5.15.2.1.7.

See also: 4.0-17.

-17 • Informative Labels -17

In labeling data fields, employ descriptive wording, or else standard, predefined terms, codes and/or abbreviations; avoid arbitrary codes.

Example: Employ descriptive labels such as STANDARD vs. MODIFIED, rather than abstract codes such as SET A vs. SET B; MALE and FEMALE, rather than GROUP 1 and GROUP 2.

Example:

(Good)

WEEK: __ MONTH: __ __ YEAR: __ __

SOCIAL SECURITY NUMBER: __ __ __ - __ __ - __ __ __ __

(Bad)

DATECODE: __ __ __ __ __

SSAN: __ __ __ __ __ __ __ __ __

Example: [See sample displays at the end of this section.]

Comment: Do not create new jargon. If in doubt, pretest all proposed wording with a sample of qualified users.

Reference: BB 2.1.6; PR 4.5.6.

See also: 2.0-11, 4.0-11.

-15 o Consistent Label Format -15

When data fields are distributed across a display, adopt a consistent format for relating labels to delineated entry areas.

Example: The label might always be to the left of the field; or the label might always be immediately above and left-justified with the beginning of the field.

Comment: Such consistent practice will help the user distinguish labels from data in distributed displays.

See also: 4.0-7.

-16 o Label Punctuation as Entry Cue -16

The label for each entry field should end with a special symbol, signifying that an entry may be made.

Example: A colon is recommended for this purpose, e.g.,

NAME: _ _ _ _ _ _ _ _ _ _ _ _

Example: [See sample displays at the end of this section.]

Comment: Choose a symbol that can be reserved exclusively for prompting user entries, or at least is rarely used for any other purpose.

Reference: BB 2.5.

See also: 4.4-11.

-13 • Explicit Tabbing to Data Fields -13

Require users to take explicit keying ("tabbing") action to move from one data entry field to the next; the computer should not provide such tabbing automatically.

Example: [See sample displays at the end of this section.]

Comment: Automatic tabbing may cause cascading of errors, if a skilled typist keys a series of items without looking at the display and has accidentally overrun one of the earlier data fields. An acceptable solution here is to design each field to end with an extra (blank) character space; software should be designed to prevent keying into a blank space, and an auditory signal should be provided to alert the user when that is attempted. This will permit consistent use of tab keying by the user to move accurately from one field to the next, even for touch typists.

Reference: MS 5.15.4.3.6; PR 4.9.1.

See also: 1.1-22.

-14 • Distinctive Label Format -14

Make labels for data fields distinctive, so that they will not be readily confused with data entries, labeled control options, guidance messages, or other displayed material.

Example: Labels might be displayed in capital letters always followed by a colon. Or labels might be displayed in dim characters, with data entries brighter.

Example: [See sample displays at the end of this section.]

Reference: BB 2.1.1; MS 5.15.4.3.5; PR 3.3.2.

See also: 1.4-16, 2.1.2-7, 4.0-8.

-11 o Marking Required/Optional Data Fields -11

Form displays should clearly and consistently distinguish between required and optional entry fields.

Example: Field delineation cues may be used for this purpose, perhaps a broken underscore to indicate required entries and a dotted underscore to indicate optional entries:

LICENSE NUMBER: _ _ _ _ _ _

MAKE: . . . . . .

YEAR/MODEL: . . . . . .

Example: Alternatively, it might be preferable to distinguish required vs. optional entry fields by coding their labels, perhaps displaying in parentheses the labels of optional fields.

Reference: BB 2.6; MS 5.15.4.3.4; PR 4.8.6.

See also: 4.4-11.

-12 • Automatic Justification of Variable-Length Entries -12

When item length is variable, computer processing should provide automatic justification; a user should not have to justify an entry either right or left, and should not have to remove any unused underscores.

Reference: BB 2.2.2; EG 6.3.2; MS 5.15.4.3.9.

-9 • Marking Field Boundaries -9

Display special characters or other consistent means of highlighting to clearly delineate each data field.

Example: Brightness inversion ("inverse video") is a good choice for this purpose, where that capability is available; otherwise, a broken-line underscore is recommended.

Example: (Good) Enter account number: _ _ _ _ _

(Bad) Enter account number:

Example: [See sample displays at the end of this section.]

Comment: Such implicit prompts help reduce data entry errors by the user.

Reference: BB 2.2.1; EG 6.3, 6.3.1; MS 5.15.4.3.4; PR 4.8.1; Savage, Habinek and Blackstad, 1982.

See also: 1.0-5, 2.1.2-2, 4.4-11.

-10 o Prompting Field Length -10

Field delineation should provide cues to indicate when a fixed or maximum length is specified for a data entry.

Example: (Good) Enter ID: _ _ _ _ _ _ _ _ _ _ _

(Bad) Enter ID (11 characters):

Example: [See sample displays at the end of this section.]

Comment: Prompting by delineation is more effective than simply telling the user how long an entry should be. In the example cited here, underscoring gives a direct visual cue as to the number of characters to be entered, and the user does not have to count them.

Comment: Similar implicit cues should be provided when data entry is prompted by auditory displays. Tone codes can be used to indicate the type and length of data entries.

Reference: BB 2.2.1; EG 6.3; MS 5.15.4.3.7; PR 4.8.2; Smith and Goodwin, 1970.

See also: 4.4-11.

-6 o Consistent Labeling -6

Make field labels consistent; always employ the same label to indicate the same kind of data entry.

Example: A field labeled NAME should always require name entry, and not sometimes require something different like elevation.

Example: [See sample displays at the end of this section.]

-7 o Protected Labels -7

Protect field labels from keyed entry, by making the cursor skip over them automatically when a user is spacing or tabbing.

Exception: When a user must change a displayed form, including changes to field labels, then that user must be able to override label protection.

Reference: BB 1.8.13; PR 3.3.2, 4.8.1.

See also: 1.1-23, 2.0-9, 6.2-5, 6.3-3.

-8 o Labels Close to Data Fields -8

Ensure that labels are sufficiently close to be associated with their proper data fields, but are separated from data fields by at least one space.

Reference: BB 1.9.5; EG 2.3.8.

See also: 2.1.2-9.

-3 • Minimal Use of Delimiters -3

Whenever possible, allow entry of multiple data items without keying special separator or delimiter characters, either by keying into predefined entry fields or by separating sequentially keyed items with blank spaces.

Example: [See sample displays at the end of this section.]

Comment: When data items contain internal blanks, design the entry fields with a predefined structure so that users will not have to key internal delimiters.

-4 o Standard Delimiter Character -4

When a field delimiter must be used for data entry, adopt a standard character to be employed consistently for that purpose.

Example: A slash (/) may be a good choice.

Comment: Choose the special delimiter character so that it does not require shift keying.

See also: 1.0-23.

-5 • Data Field Labels -5

For each data field, display an associated label to help users understand what entries may be made.

Example: (Good) NAME: _ _ _ _ _ _ _ _ _ _ _ _ _ _

ORGANIZATION: _ _ _/_ _ _

PHONE: _ _ _-_ _ _ _

(Bad) NAME, ORGANIZATION AND PHONE

_ _ _ _ _ _ _ _ _ _ _ _ _ _

_ _ _ _

_ _ _ _ _ _ _

Reference: BB 2.1.7.

See also: 1.0-23, 4.0-11.

Data forms permit entry of predefined items into labeled fields of specially formatted displays.

-1 • Single Entry of Related Data -1

In a form-filling dialogue, when a user is entering logically related items, require just one explicit entry action at the end of the transaction sequence, rather than separate entry of each item.

Comment: Depending on form design, this practice might involve entering the entire form, or entry by page or section of a longer form. Form design should indicate to users just where explicit entry is required.

Comment: Single entry of grouped data will generally permit faster input than item-by-item entry, and should prove more accurate as well. This practice permits user review and possible data correction prior to entry, and also helps the user understand at what point grouped data are processed. It will also permit efficient cross validation of related data items by the computer.

See also: 1.0-8, 6.3-9, 6.3-18.

-2 • Flexible Interrupt -2

When multiple data items are entered as a single transaction, as in form filling, allow the user to RESTART, CANCEL, or BACKUP and change any item before taking a final ENTER action.

Reference: BB 2.10; Foley and Wallace, 1974.

See also: 1.0-8, 3.3-3 thru -5, 3.5-2, 6.0-6, 6.3-10.

Tables permit data entry and display in row-column format, facilitating comparison of related data sets.

-1 • Tables for Related Data Sets -1

When sets of data items must be entered sequentially, in a repetitive series, provide a tabular display format where data sets can be keyed row by row.

Exception: When the items in each data set exceed the capacity of a single row, tabular entry will usually not be desirable, unless there is a simple means for horizontal scrolling.

Comment: Row-by-row entry facilitates comparison of related data items, and permits potential use of a DITTO key for easy duplication of repeated entries.

Reference: PR 4.8.4.

See also: 2.6-4.

-2 • Distinctive Labels -2

Format column headers and row labels distinctively, so that users can distinguish them from data entries.

See also: 4.0-8.

-3 o Informative Labels -3

Ensure that column headers and row labels are worded informatively, so that they will help guide data entry.

See also: 4.0-11.

-4 • Tabbing within Rows -4

During tabular data entry, allow users to tab directly from one data field to the next, so that the cursor can move freely back and forth within a row (i.e., across columns).

Reference: MS 5.15.3.8.4.3.

-5 o Tabbing within Columns -5

During tabular data entry, allow users to tab directly from one data field to the next, so that the cursor can move freely up and down a column (i.e., across rows).

Reference: MS 5.15.3.8.4.3.

-6 • Automatic Justification of Entries -6

The computer should automatically handle justification of tabular data entries; a user should not have to enter blanks or other extraneous formatting characters to achieve proper justification.

Example: If a user enters "56" in a field four characters long, the system should not interpret "56 __ __" as "5600".

Reference: MS 5.15.2.2.5.

See also: 1.0-29.

-7 o Justification of Numeric Entries -7

During tabular data entry, allow users to make numeric entries without concern for justification; the computer should automatically justify numeric data with respect to a fixed decimal point.

Example: A dollars-and-cents entry made at the beginning of a field (14.37 __ __) should automatically be justified to the right (__ __ 14.37) when later displayed.

Reference: PR 4.8.10.

-8 • Aiding Entry of Duplicative Data -8

For entry of tabular data, when entries are frequently repeated, provide users with some easy means to copy duplicated data.

Example: Perhaps a DITTO key should be provided.

Comment: A "ditto" capability will speed data entry, and should prove more accurate than requiring users to rekey duplicated data.

-9 • Row Scanning Cues -9

For dense tables, those with many row entries, provide some extra visual cue to guide a user accurately across columns.

Example: A blank line after every fifth row is recommended. Alternatively, adding dots between columns at every fifth row may suffice.

Example: As an alternative, provide a displayed ruler which a user can move from one row to another.

Comment: Visual aids for scanning rows are probably needed more when a user is reviewing and changing displayed data than for initial data entry. Such aids should be provided consistently, however, so that display formats for both data entry and review will be compatible.

See also: 2.1.3-12.

Interactive graphics permit convenient entry and change of data representing spatial or other functional relations.

(Guidelines for graphic data entry are deferred pending further review.)

Data Validation refers to checking entries for incorrect content and/or format, as defined by software logic.

-1 • Automatic Data Validation -1

Provide software for automatic data validation to check any item whose entry and/or correct format or content is required for subsequent data processing.

Example: If a date is entered as "February 31", the computer should generate an error message asking for a revised entry.

Comment: Do not rely on a user always to make correct entries. Computer aids for checking data entries will improve accuracy.

Comment: Some data entries, of course, may not need checking, or may not be susceptible to computer checking, such as free text entries in a COMMENT field.

Reference: MS 5.15.2.1.5; PR 4.12.4.

See also: 6.3-17, 6.3-18.

-2 • Accepting Correct Entries -2

Ensure that every possible correct data entry will be accepted and processed properly by the computer.

Example: As a negative example, on 1 June 1983, after several previous months of successful use, the computers controlling Massachusetts automobile emission inspections failed; it was discovered that they would not accept a "June" entry.

Comment: This guideline states the obvious, and might seem unnecessary except for occasional design lapses such as that cited in the example.

-3 • Non-Disruptive Error Messages -3

If data validation detects a probable error, display an error message to the user at the completion of data entry; do not interrupt an on-going transaction.

See also: 1.0-8, 4.3-10.

-4 • Deferral of Required Data Entry -4

If a user wishes to defer entry of a required data item, require the user to enter a special symbol in the data field to indicate that the item has been temporarily omitted rather than ignored.

Reference: MS 5.15.4.3.11; PR 4.8.7, 4.12.2.

See also: 4.0-2.

-5 o Reminder of Deferred Entry -5

When a user has not entered required data, but has deferred entry instead, data validation software should signal that omission to the user, permitting either immediate or delayed entry of missing items.

Reference: BB 5.2.4; MS 5.15.4.3.11, 5.15.7.5.c; PR 4.8.7.

-6 • Timely Validation of Sequential Transactions -6

In a repetitive data entry task, validate the data for one transaction and allow the user to correct errors, before permitting the user to begin another transaction.

Comment: This is particularly important when the task requires transcription from source documents, so that a user can detect and correct entry errors while the relevant document is still at hand.

See also: 3.5-12, 6.3-11.

-7 • Optional Item-by-Item Validation -7

For novice users, consider providing optional item-by-item data validation within a multiple-entry transaction.

Comment: That capability, which might be termed an "interim ENTER", may sometimes help a novice user who is uncertain about the requirements imposed on each data item. But it will slow a skilled user if item-by-item processing introduces any delay. Providing such a capability as an optional feature would help novices without hindering more experienced users.

Reference: EG 6.3.9, 7.1.

See also: 4.3-10.

Other kinds of computer processing may be provided to facilitate data entry.

-1 • Default Values -1

Provide default values to speed data entry, when those can be defined for a particular task.

-2 o User Definition of Default Values -2

When interface designers cannot predict what default values will be helpful, which is often the case, permit users (or perhaps some authorized supervisor) to define, change or remove default values for any data entry field.

Reference: MS 5.15.6.8.

-3 o Display of Default Values -3

On initiation of a data entry transaction, display currently defined default values in their appropriate data fields.

Comment: Do not expect users to remember them.

Comment: It may be helpful to mark or highlight default values in some way to distinguish them from new data entries.

Reference: BB 2.1.10; MS 5.15.6.7.

See also: 4.4-7, 6.3-7.

-4 o Easy Confirmation to Enter Default Values -4

Provide users with some simple means to confirm acceptance of a displayed default value for entry.

Example: Simply tabbing past the default field may suffice.

Comment: Employ similar techniques for user review of previously entered data.

Reference: MS 5.15.6.7, 5.15.6.10.

See also: 6.3-7.

-5 o Temporary Replacement of Default Values -5

Allow users to replace any data entry default value with a different entry, without thereby changing the default definition for subsequent transactions.

Reference: MS 5.15.6.9.

-6 • Automatic Generation of Routine Data -6

The computer should enter routine data automatically, i.e., data that can be determined from existing records.

Example: Do not require a user to identify a work station to initiate a transaction, nor to include other routine data such as current date and transaction sequence codes.

Exception: Some data entry routines may be imposed in the interest of security, but at the risk of hindering the user in achieving effective task performance. Other means of ensuring data security should be considered.

Reference: BB 2.4.2; MS 5.15.2.1.6.

See also: 6.1-1, 6.3-16.

-7 • Automatic Computation of Derived Data -7

Provide automatic computation of derived data, so that a user does not have to calculate and enter any number that can be derived from data already accessible to the computer.

Example: Statistical descriptors such as sums, means, etc., can all be derived automatically by appropriate software.

Reference: MS 5.15.2.1.6.

See also: 6.3-16.

-8 • User Review of Prior Entries -8

When data entries made in one transaction are relevant to a subsequent transaction, the computer should retrieve and display them for user review if appropriate.

Comment: Do not require a user to re-enter such data.

Reference: BB 2.4.2.

See also: 6.3-15.

-9 • Automatic Entry of Redundant Data -8

The computer should retrieve and enter redundant data items automatically, i.e., data accessible to the computer that are logically related to other entries.

Example: A user should not have to enter both an item name and identification code when either one defines the other.

Exception: Redundant entry may be needed for resolving ambiguous entries, for user training, or for security (e.g., user identification).

Comment: When verification of previously entered data is required, ask users to review and confirm data items rather than re-enter them.

Reference: BB 2.4.2, 4.3.6; EG 6.3.10; MS 5.15.2.1.6.

See also: 6.3-16.

-10 • Automatic Cross-File Updating -10

Provide automatic cross-file updating whenever necessary, so that a user does not have to enter the same data twice.

Example: When an aircraft has been assigned to a mission, the computer should automatically update both aircraft status and mission assignment files to indicate that commitment.

Reference: MS 5.15.2.1.6.

See also: 6.3-16.

Changes to the software design of data entry functions may be needed to meet changing operational requirements.

-1 • Flexible Design for Data Entry -1

When data entry requirements may change, which is often the case, provide some means for users (or a system administrator) to make necessary changes to data entry functions.

Comment: Data entry functions that may need to be changed are those represented in these guidelines, including changes to procedures, entry formats, data validation logic, and other associated data processing.

Comment: Many of the preceding guidelines in this section imply a need for design flexibility. Much of that needed flexibility can be provided in initial interface design. Some guidelines, however, suggest a possible need for subsequent design change, and those guidelines are cited below.

See also: 1.0-4, 1.0-6, 1.3-22, 1.4-23, 1.7-1, 1.8-2, 1.8-7, 1.8-9, 1.8-10.

SECTION 2

DATA DISPLAY

Data display refers to computer output of data to a user, and assimilation of information from such outputs. Some kind of display output is needed for all information handling tasks. Data display is emphasized particularly in monitoring and control tasks. Included as data display may be hardcopy printouts as well as more mutable electronic displays. Also included are auxiliary displays and signaling devices, including voice output, which may alert users to unusual conditions.

In this discussion, data are considered to be display elements related to a user's information handling task. Displayed data might consist of stock market quotations, or the current position of monitored aircraft, or a page of textual information, or a message from another user. Displayed data might provide guidance to a user in performing a maintenance task, or might provide instruction to a user who is trying to learn mathematics or history.

There may be a number of other display elements that do not constitute task-related data. Those elements include labels, prompts, computer-generated advisory messages, and other information that helps a user interact with a computer system. Although such process-related display elements are sometimes mentioned in the guidelines presented here for data display, they are discussed more extensively in a later section concerned with user guidance.

In general, less is known about data display, and information assimilation by the user, than about data entry. In current information system design, display formatting is an art. Guidelines are surely needed. But those guidelines will simply serve to help a designer become more proficient in the art.

It must be recognized, however, that guidelines cannot tell a designer what the specific contents of a display should be, but only how those contents should be presented. The specific data that must be displayed can only be determined through a careful task analysis to define the user's information requirements.

For effective task performance, displayed data must be relevant to the user's needs. An early statement of the need for relevance in data display, although written before common adoption of gender-free wording, otherwise seems valid still:

> When we examine the process of man-computer communication from the human point of view, it is useful to make explicit a distinction which might be described as contrasting "information" with "data." Used in this sense, information can be regarded as the answer to a question, whereas data are the raw materials from which information is extracted. A man's questions may be vague, such as, "What's going on here?" or "What should I do now?" Or they may be much more specific. But if the data presented to him are not relevant to some explicit or implicit question, they will be meaningless. . . .
>
> What the computer can actually provide the man are displays of data. What information he is able to extract from those displays is indicated by his responses. How effectively the data are processed, organized, and arranged prior to presentation will determine how effectively he can and will extract the information he requires from his display. Too frequently these two terms data and information are confused, and the statement, "I need more information," is assumed to mean, "I want more symbols." The reason for the statement, usually, is that the required information is not being extracted from the data. Unless the confusion between data and information is removed, attempts to increase information in a display are directed at obtaining more data, and the trouble is exaggerated rather than relieved.
>
> (Smith, 1963b, pages 296-297)

Certainly this distinction between data and information should be familiar to psychologists, who must customarily distinguish between a physical stimulus (e.g., "intensity" of a light) and its perceived effect ("brightness"). The distinction is not familiar to system designers, however, although the issue itself is often addressed. In the following description of what has been called the "information explosion", notice how the terms data and information are used interchangeably, confounding an otherwise incisive and lively analysis:

> The sum total of human knowledge changed very slowly prior to the relatively recent beginnings of scientific thought. But it has been estimated that by 1800 it was doubling every 50 years; by 1950, doubling every 10 years; and by 1970, doubling every 5 years. . . . This is a much greater growth rate than an exponential increase. In many fields, even one as old as medicine, more reports have been written in the last 20 years than in all prior human

history. And now the use of the computer vastly multiplies the rate at which information can be generated. The weight of the drawings of a jet plane is greater than the weight of the plane. The computer files of current IBM customer orders contain more than 100 billion bits of information -- more than the information in a library of 50,000 books.

For man, this is a hostile environment. His mind could no more cope with this deluge of data, than his body could cope with outer space. He needs protection. The computer -- in part the cause of the problem -- is also the solution to the problem. The computer will insulate man from the raging torrents of information that are descending upon him.

The information of the computerized society will be gathered, indexed, and stored in vast data banks by the computers. When man needs a small item of information he will request it from the computers. The machines, to satisfy his need, will sometimes carry on a simple dialogue with him until he obtains the data he wants. With the early computers, a manager would often have dumped on his desk an indigestible printout -- sometimes several hundred pages long. Now the manager is more likely to request information when he needs it, and receive data about a single item or situation on a screen or small printer.

It is as though man were surviving in the depths of this sea of information in a bathyscaphe. Life in the bathyscaphe is simple, protected as it is from the pressure of the vast quantities of data. Every now and then man peers through the windows of the bathyscaphe to obtain facts that are necessary for some purpose or other. The facts that he obtains at any one time are no more than his animal brain can handle. The information windows must be designed so that man, with his limited capabilities, can locate the data he wants and obtain simple answers to questions that may need complex processing.

(Martin, 1973, page 6)

For data display, as in other areas of user interface design, some general concepts deserve emphasis, including the importance of context, consistency, and flexibility. Somehow a means must be found to provide and maintain context in data displays so that a user can find needed information. Task analysis may point the way

here, indicating what data are relevant to each step in task performance. Design guidelines must emphasize the value of displaying no more data than the user needs, and the importance of maintaining consistent display formats so that the user always knows where to look for different kinds of information, on any one display and from one display to another.

Detailed user information requirements will vary from time to time, however, and may not be completely predictable in advance, even from a careful task analysis. Here is where flexibility is needed, so that data displays can be tailored on-line to user needs. Such flexibility is sometimes provided through optional category selection, and display offset and expansion features. If such options for tailoring display coverage are available, a user may be able to adjust the assimilation of displayed data in a way analogous to self pacing of data inputs.

In tasks where a user must both enter and retrieve data, which is often the case, the formatting of data displays should be consistent with the methods used for data entry. As an example, if data entry is accomplished via a form-filling dialogue, with a particular format for data fields, subsequent retrieval of that data set should produce an output display with the same format, especially if the user is expected to make changes to displayed data, and/or additional entries. Where compaction of data output is required for greater efficiency, perhaps to review multiple data sets in a single display frame, the displayed items should retain at least an ordering and labeling compatible with those fields used previously for data entry.

Display design should also be compatible with dialogue types used for sequence control, and with hardware capabilities. Where user inputs are made via menu selection, using a pointing device like a lightpen, then display formats should give prominence (and adequate separation) to the labeled, lightpennable options. Location of multi-function keys at the display margin, to be labeled on the adjacent portion of the display itself, may provide flexibility for both data entry and sequence control, but will necessarily constrain display formatting.

These general concepts underlie many of the specific guidelines for data display proposed in the following pages. As in the other areas of user interface design, an attempt has been made to write guidelines for data display in functional terms, insofar as possible without reference to specific display devices and questions of hardware implementation. As a practical matter, however, available display technology will inevitably influence the wording of guidelines in some degree.

A discerning reader will note that the guidelines presented here deal almost exclusively with visual displays; there are only a few references to other possible display modes. Moreover, most of these guidelines implicitly assume a fairly large visual display, with room to show different kinds of data at one time -- in effect, a display with about 24 lines of 80 characters, much like the devices we now use. Consequently, many of these guidelines will not apply in applications where displays are constrained to a smaller size, such as "briefcase" terminals or handheld devices.

As display technology develops further, it seems inevitable that some of the guidelines proposed here must be reconsidered, and other guidelines added. As an example, we may anticipate increased use of graphics in future information display design, with moving (and talking) pictures such as those we now enjoy in displays designed for entertainment. Guidelines for graphic display have been drafted, but those are still undergoing critical review and are not included here.

The guidelines proposed here are intended for display designers. If we regard displays as contrived arrangements of data, then the guidelines refer to that contrivance. What happens in applications where the computer provides a flexible capability allowing users to contrive many of their own displays? If a user composes a poor page of text, with run-on sentences, flawed grammar, inconsistent spacing, etc., can a software designer be held responsible? Presumably not, or at least not today.

One might imagine future systems, however, where some form of expertise is stored in the computer, including expertise on user interface design. In applications where users design their own displays, a computer might someday be prepared to offer guidelines on the subject, or even to enforce design rules where warranted. If a user entered irregularly spaced text, a computer might regularize the spacing according to some consistent rule in subsequent output of that text.

The guidelines presented here can themselves be regarded as a long multipage data display. Problems of display organization arise in presenting the guidelines material, in terms of content, wording, and format. The topical organization of these guidelines is based on function, dealing with different types of displayed data and display manipulation. For some topics, however, it is not clear where they should be covered. For example, guidelines pertaining to "lists" are included here with guidelines for text display, although that topic is also related to tabular displays. Some broadly applicable topics are presented in a general introductory section.

- Data display refers to computer output of data to a user, and assimilation of information from such outputs.
- Data displays can output text, forms, tables, and graphics, or combinations of those various types of data.

- Density refers to how much data must be output in any display; ideally, provide just what is needed and no more.
- Format refers to organization of different types of data in a display to aid assimilation of intormation.
- Coding refers to distinctive means for highlighting different categories of displayed data for user attention.
- Display generation refers to the means for specification of data outputs, either by a user or automatically.
- Framing refers to user control of data coverage by display movement, including paging, scrolling, offset, etc.
- Display update refers to regeneration of changed data to show current status, by user request or automatically.
- Suppression refers to user control of display coverage by temporary deletion of specified data categories.
- Changes to the software design of data display functions may be needed to meet changing operational requirements.

Objectives:

Consistency of data display
Efficient information assimilation by the user
Minimal memory load on user
Compatibility of data display with data entry
Flexibility for user control of data display

Guidelines:

Data display refers to computer output of data to a user, and assimilation of information from such outputs.

-1 • Necessary Data Displayed -1

At any step in a transaction sequence, ensure that whatever data a user needs will be available for display.

Example: Even temporary loss of needed data, as might be caused by display blanking during automatic data update, is not acceptable in many design applications.

Comment: The designer of user interface software must employ some method of task analysis (e.g., operational sequence diagrams) in order to determine a user's detailed information requirements.

Reference: BB 4.3.6.

See also: 2.0-2, 2.2-1, 4.0-5.

-2 o Only Necessary Data Displayed -2

Tailor the display of data to user needs, providing only necessary and immediately usable data at any step in a transaction sequence.

Comment: Display of extraneous data may confuse a user and slow assimilation of needed information.

Comment: When user information requirements cannot be exactly anticipated by the designer, which is sometimes the case, provision should be made for on-line tailoring of displays by the user, including data selection (Section 2.5), data coverage within a display frame (Section 2.6) and the suppression of displayed data (Section 2.8).

Reference: BB 1.7, 1.8.10; EG 3.1.4, 3.3.1; MS 5.15.3.1.2, 5.15.4.6.2; Stewart, 1980.

See also: 2.0-1, 2.0-7, 2.2-1, 2.2-2, 2.9-1, 4.0-5.

-3 • Data Displayed in Usable Form -3

Display data to users in directly usable form; do not make users convert displayed data.

Example: If distance (or altitude) might be required in either meters or feet, then display both values.

Example: This recommendation applies to error messages and other forms of user guidance as well as to data displays.

(Probably adequate) Character in NAME entry cannot be recognized.

(Too cryptic) Error 459 in column 64.

Comment: Do not require a user to transpose, compute, interpolate, or translate displayed data into other units, or refer to documentation to determine the meaning of displayed data.

Reference: BB 3.3; EG 3.3.4; MS 5.15.3.1.3.

See also: 4.4-1.

-4 • Data Display Consistent with User Conventions -4

Display data consistent with standards and conventions familiar to users.

Example: If users work with metric units of measurement, do not display data in English units.

Example: Computer time records that are not in directly usable format should be converted for display, to a conventional 12-hour (AM/PM) clock or a 24-hour clock, in local time or whatever other time standard is appropriate to user needs.

Example: Calendar formats should follow user customs.

(American calendar)

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

(European calendar)

S	1	8	15	22	29
M	2	9	16	23	30
T	3	10	17	24	31
W	4	11	18	25	
T	5	12	19	26	
F	6	13	20	27	
S	7	14	21	28	

Reference: BB 3.4; EG 2.2.4.

See also: 4.0-16.

-5 o Establishing Display Standards -5

When no specific user conventions have been established, adopt some consistent data display standards for user interface design.

-6 • Consistent Display Format -6

For any particular type of data display, maintain consistent format from one display to another.

Comment: When an item is missing from a standard format, display that item as a labeled blank rather than omitting it altogether.

Reference: BB 1.1.1; MS 5.15.3.2.1; Stewart, 1980.

-7 • User Control of Data Display -7

Allow users to control the amount, format, and complexity of displayed data, as necessary to meet user needs and task requirements.

Comment: An experienced user may be able to deal with more complex displays than a novice. But a user experienced in one task may be a novice in another. Thus a range of display tailoring capabilities may be desirable for any particular task.

Comment: Increasing the options for user control of data displays will add somewhat to what a new user must learn about a system, and so will involve a trade-off against simplicity of user interface design.

Reference: EG 3.4.2.

See also: 2.0-2, 2.5-1, 2.9-1.

-8 o User Changes to Displayed Data -8

Allow users to change displayed data, or enter new data, when that represents a functionally efficient transaction sequence.

Comment: In many displays, of course, it is not feasible or desirable for users to change data, such as in operations monitoring (process control) displays, or in displays permitting access to a large, stable data base.

Comment: Some consistent formatting cue, perhaps different cursor shape or different initial cursor placement, should be provided to inform the user when displayed data can or cannot be changed.

Reference: PR 4.4.

See also: 1.0-5, 1.3-2, 6.2-3.

-9 o Protection of Displayed Data -9

When protection of displayed data is essential, maintain computer control over the display and do not permit a user to change controlled items.

Comment: Never assume compliance with instructions by the user, who may attempt unwanted changes by mistake, or for curiosity, or to subvert the system.

Reference: EG 3.4.8.

See also: 1.1-23, 1.4-7, 6.2-4, 6.3-3.

-10 • Context for Displayed Data -10

Each data display should provide needed context, recapitulating prior data as necessary so that a user does not have to rely on memory to interpret new data.

Comment: When user information requirements cannot be determined in advance, which is sometimes the case, it may be desirable to provide a separate display window as a "notepad" in which a user can preserve needed items by marking those to be saved.

Comment: If data must be remembered from one display to another, display no more than four to six items.

Reference: BB 4.3.6; EG 2.3.15.

-11 • Familiar Wording -11

The wording of displayed data and labels should incorporate familiar terms and the technical jargon of the users, and avoid the unfamiliar jargon of designers and programmers.

Comment: When in doubt, pretest the meaning of words for prospective users, to ensure that there is no ambiguity.

Reference: BB 3.7.1, 3.7.4; EG 3.4.5, 4.2.13; PR 4.5.6.

See also: 1.4-17, 4.0-16, 4.0-17, 4.3-3.

-12 o Consistent Wording -12

For displayed data and labels, choose words carefully and then use them consistently.

Example: The word "screen" should not be used to mean "display frame" in one place, and "menu selection option" in anotner.

Comment: Consistent word usage is particularly important for technical terms. Standard terminology should be defined and documented in a glossary for reference by interface designers and by users.

Reference: BB 1.2.2, 3.7.2; EG 3.4.5, 4.2.13.

-13 o Consistent Wording Across Displays -13

Ensure that wording is consistent from one display to another.

Example: The title of a display should be identical to the menu option used to request that display.

Reference: BB 3.7.4.

-14 • Minimal Use of Abbreviation -14

Display complete words in preference to abbreviations.

Exception: Abbreviations may be displayed if they are significantly shorter, save needed space, and will be understood by the prospective users.

Exception: When abbreviations are required (or useful) for data entry, then corresponding use of those abbreviations in data display may help a user learn them for data entry.

Reference: BB 3.1, 3.1.1, 3.1.5; EG 4.1.3; MS 5.15.3.2.3.

-15 o Consistent Abbreviation -15

Adopt a consistent form for all abbreviations.

Comment: If an abbreviation deviates from the consistent form, it may be helpful to give it some special mark whenever it is displayed.

Reference: BB 3.1.2; MS 5.15.3.2.3; PR 4.5.6; Moses and Ehrenreich, 1981.

See also: 1.0-16 thru 1.0-22.

-16 o Distinctive Abbreviations -16

Ensure that abbreviations are distinctive, so that abbreviations for different words are distinguishable.

Reference: BB 3.1; MS 5.15.3.2.3; Moses and Ehrenreich, 1981.

-17 o Dictionary of Abbreviations -17

When abbreviations are used, provide a dictionary of abbreviations available for on-line user reference.

Reference: BB 3.1.3; MS 5.15.3.2.3.

See also: 4.4-16.

-7 o Consistent Label Location -7

Place each label in a consistent location, adjacent to (above, or to the left of) the labeled data item (or group of items).

Example: In a numbered list, vertically formatted, the numeric labels should be aligned so that the data items start in a fixed column position on the display.

Comment: Consistent alignment of labels and data will aid display scanning by a user.

Reference: EG 2.3.9; MS 5.15.3.1.10.a.

See also: Section 2.1.3.

-8 o Distinctive Label Format -8

Labels should be distinctive in format/positioning to help users distinguish them from displayed data and other types of displayed material (e.g., error messages).

Example: Labels might be all upper case when data are displayed in mixed case, or might be displayed as dimmed when data are bright, or might perhaps be displayed in a different font where that capability exists.

Example: [See sample displays at the end of this section.]

Reference: EG 3.2.3; MS 5.15.3.1.10.c, 5.15.4.3.5.

See also: 4.0-8.

-9 o Labels Close to Data Fields -9

Ensure that labels are sufficiently close to be associated with their data fields, but are separated from their data fields by at least one space.

Reference: BB 1.9.5; EG 2.3.8.

See also: 1.4-8.

-4 o Descriptive Wording of Labels -4

The word or phrase that labels a data field should describe the data content.

Example: [See sample displays at the end of this section.]

Comment: Labels should be worded carefully so that they assist users in scanning the display and assimilating information quickly.

Reference: BB 3.5; EG 3.2; MS 5.15.3.1.10.

-5 o Consistent Wording of Labels -5

Ensure that labels are worded consistently, so that the same data item is given the same label if it appears in different displayed forms.

Example: [See sample displays at the end of this section.]

Comment: It may also help to employ consistent grammatical format for different labels; i.e., do not use single words or phrases for some labels and short sentences for others, or use verbs for some and nouns for others.

Reference: BB 3.8.4; MS 5.15.3.1.6.

See also: 4.0-21.

-6 o Distinctive Wording of Labels -6

Ensure that field labels are distinctive from one another in wording, to aid user discrimination.

Reference: BB 3.5; EG 3.2.3.

Data forms display sets of related data items in defined fields, formatted and labeled to aid data entry and review.

-1 • Forms for Related Data -1

Use forms to display related sets of data items in separately labeled fields.

Comment: Forms can aid review of related data items by displaying explanatory labels to caption each data field.

-2 • Visually Distinctive Data Fields -2

Display formats should provide clear visual definition of different data fields.

Example: [See sample displays at the end of this section.]

Reference: MS 5.15.4.3.4.

See also: 1.0-5, 1.4-9.

-3 o Data Field Labeling -3

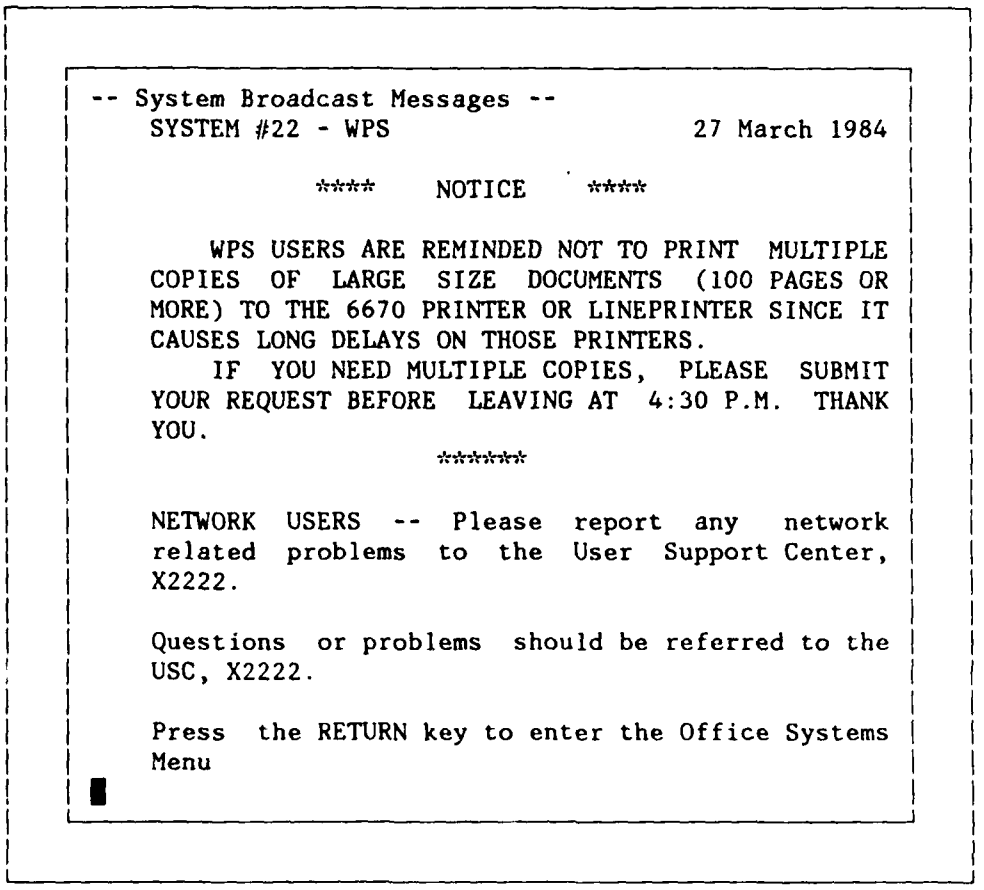
Identify each data field with a displayed label.

Comment: Do not assume that the user can identify individual data fields because of past familiarity. Context may play a significant role: 617-271-7768 might be recognized as a telephone number if seen in a telephone directory, but might not be recognized as such in an unlabeled display.

Reference: BB 1.8.7; EG 2.2.16; MS 5.15.3.1.9.

See also: 1.4-5 thru -8, 4.0-11.

(Bad) Sample Text Display (Bad)



```
-- System Broadcast Messages --
   SYSTEM #22 - WPS                          27 March 1984

                ****    NOTICE    ****

      WPS USERS ARE REMINDED NOT TO PRINT MULTIPLE
   COPIES OF LARGE SIZE DOCUMENTS (100 PAGES OR
   MORE) TO THE 6670 PRINTER OR LINEPRINTER SINCE IT
   CAUSES LONG DELAYS ON THOSE PRINTERS.
      IF YOU NEED MULTIPLE COPIES, PLEASE SUBMIT
   YOUR REQUEST BEFORE LEAVING AT 4:30 P.M. THANK
   YOU.
                        *******

   NETWORK USERS -- Please report any network
   related problems to the User Support Center,
   X2222.

   Questions or problems should be referred to the
   USC, X2222.

   Press the RETURN key to enter the Office Systems
   Menu
```

This bad text display violates in some degree several design guidelines in this section:

2.1.1- 1 Conventional text display
- 2 Consistent text format
- 3 Mixed case for displayed text
- 4 Separation of paragraphs
- 5 Consistent Word Spacing
- 7 Conventional punctuation
- 8 Clarity of wording
-10 Simple sentence structure

(Good) Sample Text Display (Good)

SYSTEM BROADCAST MESSAGES

27 March 1984

Word Processing Users:

Please do NOT print multiple copies of large documents (more than 100 printed pages) during normal working hours. Large print requests will delay printing service for all users.

If you do need bulk printing, submit your request before leaving at 4:30 pm. Your printouts will be ready by the next morning.

Network Users:

Please report any net-related problems to the User Support Center, x2222.

* Press CONTINUE to display the Office Systems Menu.

These sample displays represent a broadcast message received by all users logging onto an on-line office support system. The wording of the bad display is taken from an actual instance. The good display clarifies wording of the text and improves display formatting.

The bad display is annotated to indicate violations of several of the design guidelines proposed here for text display. Although most of its noted deficiencies are minor, in sum they create a display that is potentially confusing to its users.

-19 o List Ordering in Multiple Columns -19

If a list is displayed in multiple columns, order the items vertically within each column.

Example:

(Good)

S. R. Abbott	B. M. Drake	K. L. Kessel
C. N. Abernethy	S. M. Dray	R. T. Kintz
C. A. Adams	M. G. Dumoff	G. A. Klein
H. L. Ammerman	R. C. Eakins	R. P. Koffler
C. J. Arbak	S. L. Ehrenreich	J. R. Kornfeld
etc.		

(Bad)

S. R. Abbott	C. N. Abernethy	C. A. Adams
H. L. Ammerman	C. J. Arbak	A. J. Aretz
A. F. Aucella	J. A. Ballas	M. C. Bardales
S. H. Barry	D. R. Baum	J. D. Beattie
W. Beavers	D. C. R. Benel	T. Berns
etc.		

-20 o Hierarchic Structure for Long Lists -20

For a long list, extending more than one displayed page, consider the possibility of adopting a hierarchic structure to permit its logical partitioning into related shorter lists.

-21 • Abbreviations Defined in Text -21

When words in text displays are abbreviated, define each abbreviation in parentheses following its first appearance.

Comment: This practice will help only those users who read displayed text from front to back, and remember what they have read. For forgetful users, and for users who sample later sections of a multipage text display, abbreviations may still seem undefined. For such users, it might be helpful to provide an on-line dictionary of abbreviations for convenient reference.

Reference: BB 3.1.8.

See also: 2.0-17, 4.4-16.

-17 o Single-Column List Format -17

Each item in a list should start on a new line, i.e., the list should be a single column.

Example: (Good) Major USI functional areas include

Data Entry
Data Display
Sequence Control
User Guidance
Data Transmission
Data Protection

(Bad) Major USI functional areas include

Data Entry	Data Display
Sequence Control	User Guidance
Data Transmission	Data Protection

Exception: Listing in multiple columns may be considered where shortage of display space dictates a compact format.

Exception: Multiple columns of data should be used where that facilitates comparison of corresponding data sets, as in tabular displays (Section 2.1.3).

Reference: BB 1.9.2; EG 2.3.5; MS 5.15.3.5.6.1.

-18 o Logical List Ordering -18

Adopt some logical principle by which to order lists; where no other principle applies, order lists alphabetically.

Comment: It is the user's logic that should prevail rather than the designer's logic (where those are different).

Reference: EG 2.3.1; MS 5.15.3.5.6.

See also: 2.1.3-6, 2.3-13 thru -17.

-14 o Active Voice -14

Compose sentences in the active rather than passive voice.

Example: (Good) Clear the screen by pressing RESET.

(Bad) The screen is cleared by pressing RESET.

Comment: Active voice sentences are generally easier to understand.

Reference: BB 3.8.5.

See also: 4.0-19

-15 o Temporal Sequence -15

When a sentence describes a sequence of events, phrase it with a corresponding word order.

Example: (Good) Enter LOGON before running programs.

(Bad) Before running programs, enter LOGON.

Comment: Temporal order is clearer. Reverse order may confuse the user.

Reference: BB 3.8.6.

See also: 4.0-20.

-16 • Lists for Related Items -16

For a series of related items (words, phrases, instructions, etc.), display those items in a list rather than in running text.

Comment: A list format will facilitate rapid, accurate scanning.

Reference: BB 1.3.2.

-11 o Concise Wording -11

When speed of display output for textual material is slower than the user's normal reading speed, an extra effort should be made to word the text concisely.

Comment: Assume a normal average reading speed of 250 words per minute.

Reference: EG 3.3.7.

See also: 4.3-5.

-12 o Distinct Wording -12

Use distinct words rather than contractions or combined forms, especially in phrases involving negation.

Example: (Good) "will not", "not complete"

(Bad) "won't", "incomplete"

Comment: This practice will help the user understand the sense of the message.

Reference: BB 3.1.4; EG 2.2.15.

-13 o Affirmative Sentences -13

Use affirmative statements rather than negative statements.

Example: (Good) Clear the screen before entering data.

(Bad) Do not enter data before clearing the screen.

Comment: Tell the user what to do, rather than what to avoid.

Reference: BB 3.8.3.

See also: 4.0-18.

-7 o Conventional Punctuation -7

Use conventional punctuation in textual display; sentences should end with a period (or question mark).

Example: [See sample displays at the end of this section.]

Reference: BB 1.3.4; EG 2.2.13.

-8 • Clarity of Wording -8

In designing text displays, especially text composed for user guidance, strive for simplicity and clarity of wording.

Example: [See sample displays at the end of this section.]

-9 o Sentences Begin with Main Topic -9

Put the main topic of each sentence near the beginning of the sentence.

Reference: BB 3.8.2.

-10 o Simple Sentence Structure -10

Use short, simple sentences.

Example: [See sample displays at the end of this section.]

Comment: Long sentences with multiple clauses may confuse the user. Consider breaking long sentences into two or more shorter statements.

Reference: BB 3.8, 3.8.1; EG 2.2.12; Wright and Reid, 1973.

See also: 4.3-5.

-4 o Separation of Paragraphs -4

Separate displayed paragraphs of text by at least one blank line.

Example: [See sample displays at the end of this section.]

Reference: EG 2.3.4; MS 5.15.3.7.3.

-5 o Consistent Word Spacing -5

Displayed text should be left-justified to maintain consistent spacing between words, leaving right margins ragged if that is the result.

Example: [See sample displays at the end of this section.]

Exception: Right justification should be employed if it can be achieved by variable spacing, maintaining constant proportional differences in spacing between and within words, and consistent spacing between words in a line.

Comment: Reading is easier with constant spacing, which outweighs the advantage of an even right margin achieved at the cost of uneven spacing. Uneven spacing is a greater problem with narrow column formats than with wide columns. Uneven spacing handicaps poor readers more than good readers.

Reference: PR 4.5.1, 4.10.5; Gregory and Poulton, 1970; Campbell, Marchetti and Mewhort, 1981.

See also: 1.3-18.

-6 o Minimal Hyphenation -6

In display of textual material, keep words intact, with minimal breaking by hyphenation between lines.

Comment: Text is more readable if the entire word is on one line, even if that makes the right margin more ragged.

Reference: BB 3.2; EG 2.2.10.

See also: 1.3-19.

Text displays permit output of stored textual data, as well as messages and other text intended for user guidance.

-1 • Conventional Text Display -1

Computer-generated displays of stored textual data, messages, or instructions, should generally follow design conventions for printed text.

Example: [See sample displays at the end of this section.]

Comment: Adoption of familiar design conventions for text display will permit users to rely on prior reading skills.

-2 • Consistent Text Format -2

When textual material is formatted, as in structured messages, adopt a consistent format from one display to another.

Example: [See sample displays at the end of this section.]

See also: 2.0-6.

-3 o Conventional Use of Mixed Case -3

Display running text (prose) conventionally, in mixed upper and lower case.

Example: [See sample displays at the end of this section.]

Exception: An item intended to attract the user's attention, such as a label or title, might be displayed in upper case.

Exception: Upper case should be used when lower case letters will have decreased legibility, which is true on display terminals that cannot show true descenders for lower case letters. (Fortunately, that limitation of display technology is fast disappearing.)

Comment: Normal reading of text is easier with conventional use of capitalization, such as to start sentences, or to indicate proper nouns and acronyms.

Reference: BB 1.6; EG 3.4.3; MS 5.15.3.7.5.

Data displays can output text, forms, tables, and graphics, or combinations of those various types of data.

-1 • Appropriate Data Types -1

Provide for display of different types of data appropriate for different user tasks.

Comment: In most applications there is a need to display textual data for labels, instructions and messages, as well as for text entry/editing. In some applications, a picture may be worth at least a thousand words, and graphics may be needed to supplement other kinds of data display. In some applications, tables and data forms may be needed. Desirable design features for display of different data types are noted in the remainder of this section.

Comment: If hardware and/or software limitations prevent display of necessary types of data, then the fundamental approach to user interface design must be reconsidered. Either system capabilities must be extended, or else user tasks must be redefined (and simplified) so that they can be performed adequately within the limited display capabilities that are available.

-18 o Minimal Punctuation of Abbreviations -18

Minimize punctuation of abbreviations and acronyms.

Example: (Good) USAF

(Bad) U.S.A.F.

Exception: Punctuation should be retained when needed for clarity, e.g., "4-in. front dimension" rather than "4 in front dimension".

Exception: Punctuation of abbreviations might be justified when an abbreviation represents more than one word, and more than the first letter of each word is included in the abbreviation (e.g., "common services" abbreviated as "COM.SER" rather than "COMSER").

Reference: BB 1.3.5; EG 2.2.14; MS 5.15.3.2.3.

-10 o Labeling Units of Measurement -10

Include the units of measurement for displayed data either in the label or as part of each data item.

Example: DISTANCE (KM): __ __ __

or

DISTANCE: __ __ __ KM

Reference: BB 1.8.8; MS 5.15.4.3.10.

See also: 1.4-19.

-11 • Consistent Format Across Displays -11

The ordering and layout of corresponding data fields should be consistent from one display to another, insofar as possible.

Reference: BB 1.8.4, 2.8.3; MS 5.15.3.1.6.

See also: 2.1.3-14.

-12 o Form Compatible for Data Entry and Display -12

When forms are used for data entry as well as for data display, the format for data display should be compatible with whatever format is used for data entry; use the same item labels and ordering for both.

See also: 1.4-22.

-13 • Consistent Format Within Data Fields -13

The detailed internal format of frequently used data fields should be consistent from one display to another.

Example: Telephone numbers should be consistently hyphenated, as 213-394-1811.

Example: Time records might be consistently formatted with colons, as HH:MM:SS, or HH:MM, or MM:SS.S, whatever is appropriate.

Example: Date records might be consistently formatted with slashes, as MM/DD/YY.

Comment: The convention chosen should be familiar to the prospective users. For European users, the formatting of telephone numbers and of dates is customarily different than suggested in the examples above. For military users, date-time data are frequently combined in a familiar special format. For many user groups, time records are kept on a 24-hour clock, which should be acknowledged in display formatting.

Reference: EG 2.2.17.

-14 • Partitioning Long Data Items -14

Divide long data items of arbitrary alphanumeric characters into subgroups of three or four characters separated by a blank (or by some special symbol).

Example: [See sample displays at the end of this section.]

Exception: Words should be displayed intact, whatever their length.

Comment: Hyphens may be used instead of blanks where that is customary. Slashes are less preferred for separating groups, since they are more easily confused with alphanumerics.

Comment: Grouping should follow convention where a common usage has been established, as in the NNN-NN-NNNN of social security numbers.

Reference: EG 2.2.2; MS 5.15.3.1.7, 5.15.3.5.8.

See also: 1.0-15.

-15 • Distinguishing Blanks from Nulls -15

For many applications, it may be desirable to distinguish blanks (keyed spaces) from nulls (no entry at all) in the display of data forms.

Example: [See sample displays at the end of this section.]

Comment: Some special symbol might be adopted to denote null entry. If field delimiters are displayed to guide data entry, then it will often be sufficient simply to leave those delimiters unchanged when no entry is made.

See also: 1.4-9.

(Good) Sample Data Form Display (Good)

```
                    VISA APPLICATION

NAME: Jones, Andrew David_______          VISA: 356 478
      LAST, FIRST MIDDLE

BIRTH COUNTRY: UK     DATE:  3/22/25
                             MM DD YY

  NATIONALITY: UK     PASSPORT: Z196284__

      ADDRESS: 5 Fairview Lane_____________
               Loughborough, LE11 3RG_______
               England______________________

OTHER TRAVELERS ON THIS VISA
                                    BIRTH
NAME:                               COUNTRY:  DATE:
Jones, Sandra Jean_____             UK        10/11/28
Jones, Cynthia Leigh_____           FR         6/12/68
_____________________________       __        __/__/__
_____________________________       __        __/__/__
LAST, FIRST MIDDLE                            MM DD YY

* Press ENTER when done.
```

These sample displays represent a possible form for entry and review of visa application data. In the good display, data entries are bolded to help distinguish them from labels and field delimiters. Fields are ordered consistently in relation to a (supposed) paper application form, and formatted to facilitate data review.

The bad display is annotated to indicate violations of several of the design guidelines proposed here for data form displays. The data entries in the bad display have been invented to suggest what a user might produce if confused by inadequate labeling and the absence of field delimiters.

(Bad) Sample Data Form Display (Bad)

```
Name Andrew D. Jones                    Visa Number 356478

Birthplace London                       Nationality English

Passport Z196284                        Birthdate Mar. 22,

Address  1925
         5 Fairview Lane, Loughborough, L
         E11 3RG, England

Other travelers on this visa
Traveler's Name                    Date of Birth - Place
Sandra J. Jones                         Oct. 11, - 1928
Birmingham
Cynthia L. Jones                        June 12, - 1968
Paris, France

Press ENTER when done
```

This bad data form display violates in some degree several design guidelines in this section:

2.1.2- 2 Visually distinctive data fields
- 4 Descriptive wording of labels
- 5 Consistent wording of labels
- 8 Distinctive label format
-14 Partitioning long data items
-15 Displaying blanks

This bad data form also violates various design guidelines pertaining to data entry, as noted at the end of Section 1.4.

Tables can display data in row-column format, to aid detailed comparison of ordered sets of items.

-1 • Tables for Data Comparison -1

When information handling requires detailed comparison of ordered sets of data, adopt a tabular format for data display.

-2 • Column and Row Labels -2

In tabular displays, label columns and rows following the same guidelines proposed for labeling the fields of data forms.

Example: [See sample displays at the end of this section.]

Reference: BB 1.8.7.

See also: Section 2.1.2.

-3 o Labeling Units of Measurement -3

In tabular displays, either consistently include the units of displayed data in the column labels, or else place them after the first row entry.

Example: (Good)

Time (s)	Velocity (m/s)	Temperature (°C)
5	8	25
21	49	29
43	87	35

(Also acceptable)

Time	Velocity	Temperature
5 s	8 m/s	25 °C
21	49	29
43	87	35

Reference: BB 1.8.8.

-4 • Justification of Numeric Data -4

Justify columns of numeric data with respect to a fixed decimal point; if there is no decimal point, then numbers should be right-justified.

Example: [See sample displays at the end of this section.]

Reference: BB 1.4.2, 1.4.3; EG 2.3.9; MS 5.15.3.5.3; PR 4.8.10, 4.10.6.

See also: 1.5-7.

-5 o Justification of Alphabetic Data -5

Left justify columns of alphabetic data to permit rapid scanning.

Example: (*Good*)

APL
COBOL
FORTRAN
PL1

(Bad)

APL
COBOL
FORTRAN
PL1

Exception: Indentation may be used to indicate subordinate elements in hierarchic lists.

Exception: A short list, of just four or five items may be displayed horizontally on a single line, in the interests of compact display format, if that is done consistently.

Reference: BB 1.3.1; EG 2.2.8, 2.2.11; MS 5.15.3.5.3.

-6 • Logical Organization -6

Organize tabular data in some recognizable order to facilitate scanning and assimilation.

Example: Dates might be ordered chronologically, names alphabetically.

Example: [See sample displays at the end of this section.]

Reference: BB 1.8.1; EG 2.2.3, 2.3.1; MS 5.15.3.1.4.

See also: 2.1.1-18, 3.1.3-19.

-7 o Tables Referenced by First Column -7

When tables with multiple columns are used for reference, display reference items, i.e., those by which the table will be accessed, in the left column; display the material most relevant for user response in the next adjacent column; and display associated but less significant material in columns further to the right.

Example: [See sample displays at the end of this section.]

Comment: As a corollary, when a list of people is ordered alphabetically by their last name, then their last names should be displayed first, i.e., as the leftmost element in each row.

Reference: Hamill, 1980.

-8 o Items Paired for Direct Comparison -8

If data items must be compared on a character-by-character basis, display one directly above the other.

Comment: But users will not be entirely accurate in making such comparisons; automated comparison by computer analysis would be more reliable.

Reference: MS 5.15.3.1.8.

-9 • Distinctive Labeling -9

When rows of tabular or listed data are labeled by number, letter, etc., those labels should be distinctively different from the labels of other listed material that might be displayed, such as menu options or instructions.

Comment: There are many ways to distinguish different types of labeled material, including consistent differences in display format/placement, special displayed markers, and/or display of pictorial symbols ("icons") to denote control options.

Reference: EG 2.2.7.

See also: 3.1.3-18.

-10 o Numbered Items Start with "1" -10

When listed data items are labeled by number, start the numbering with "1", rather than "0".

Comment: In counting, people start with "one"; in measuring, they start with "zero".

Reference: EG 2.2.6.

-11 o Repeated Elements in Hierarchic Numbering -11

For hierarchic lists with compound numbers, display the complete numbers; do not omit repeated elements.

Example: (Good)
```
2.1  Position Designation
2.1.1  arbitrary positions
2.1.1.1  discrete
2.1.1.2  continuous
2.1.2  predefined positions
2.1.2.1  HOME
2.1.2.2  other
```

(Bad)
```
2.1. Position Designation
    1. arbitrary positions
      1  discrete
      2  continuous
    2. predefined positions
      1  HOME
      2  other
```

Comment: Implicit numbering, as in the "bad" example, may be acceptable for tasks involving perception of list structure. Complete numbering is better, however, for tasks requiring search and identification of individual items in the list.

Reference: Smith and Aucella, 1983.

-12 • Row Scanning Cues -12

In dense tables with many rows, insert a blank line or some other distinctive feature at regular intervals to aid horizontal scanning.

Example: For many applications it will suffice to add a blank line after every five rows.

Example: [See sample displays at the end of this section.]

See also: 1.5-9.

-13 o Column Scanning Cues -13

When data are displayed in more than one column, separate the columns by enough blank character spaces, or some other distinctive feature, to ensure that separate entries within a row will not be confused.

Example: [See sample displays at the end of this section.]

Comment: For most tables, a column separation of at least three spaces should be maintained. Certainly the spacing between columns should be greater than any internal spaces that might be displayed within a column entry.

Reference: EG 2.3.6.

See also: 2.1.3-14.

-14 o Consistent Column Spacing -14

Column spacing should be consistent within a display, and from one display to another.

Example: [See sample displays at the end of this section.]

Reference: BB 1.8.3.

See also: 2.1.2-11, 2.1.3-13.

-15 • Consistent Label Format -15

Adopt a consistent format for labeling columns and other elements within displayed tables.

Example: Each column label might be left-justified with the leftmost character of the column entries beneath it.

Comment: Consistent left justification of labels will prove especially helpful when columns vary in width.

Reference: Hartley, Young and Burnhill, 1975.

See also: 4.0-6.

(Good) Sample Tabular Display (Good)

```
                    AUTOMOBILE OWNERS          Page 1 of 4

LICENSE       EMPLOYEE                  EXT    DEPT

MA 127 355    Michaels, Allison         7715     91
MA 135 449    Duvet, William            3898     81
MA 227 379    Smithson, Jill            2491     63
MA 227 GBH    Zadrowski, Susan          2687     53
MA 253 198    Jenskin, Erik             3687     31

MA 286 PAM    Hilsmith, Joseph          2443    100
MA 291 302    Leonard, John             7410     92
MA 297 210    Toth, Kelley              7538     45
MA 328 647    Cooksey, Roger            2144     64
MA 342 NCG    Hesen, Christopher        7544     21

MA 367 923    Maddox, Patrick           7070     66
MA 375 NRC    Ashley, Maria             3397     34
MA 376 386    Wheetley, Katherine       2638     58
MA 385 248    Malone, Frank             2144     64
MA 391 293    Lowman, Edward            8263     77

n = Next page
```

These sample displays represent a table for finding the owner of a car with a particular license plate. (Perhaps it is an employee who has parked in the wrong place, or who has left headlights burning.) In the good display, data entries are ordered by license number to aid the search.

The bad display is ordered alphabetically by employees' last name, which will not prove helpful for this task. The bad display is annotated to indicate several other violations of the design guidelines proposed here for tabular displays.

(Bad) Sample Tabular Display (Bad)

```
Automobile Owners
Sara Alwine                2438 MA 929 448 103
Christopher Aranyi         2716 MA 797 AND 97
Maria Ashley               3397 MA 375 NRC 34
Arlene Atchison            7672 NH 60731   28
Steven Bahr                3272 MA 635 203 35
David Baldwin              3295 NH 63577   75
David Benkley              3581 MA 589 ADE 58
Marlin Boudreau            3413 MA 816 HER 93
Roger Cooksey              2144 MA 328 647 64
Joseph Curran              3167 RI 4693    83
Kent Delacy                3619 MA 749 827 29
Susan Doucette             2797 MA 525 115 41
Joseph Drury               7604 NH 42265   27
William Duvet              3898 MA 135 449 81
Samuel Everett             3619 MA 635 ASK 29
Jeanne Fiske               7642 MA 614 CSU 10
Nancy Graham               2358 MA 745 CKJ 10
Paul Greenbaum             3979 MA 846 BLN 103
Christopher Hesen          7544 MA 342 NCG 21
Joseph Hilsmith            2443 MA 286 PAM 100
```

This bad tabular display violates in some degree several design guidelines in this section:

2.1.3- 2 Column and row labels
- 4 Justification of numeric data
- 6 Logical organization
- 7 Column order in reference tables
-12 Row spacing
-13 Column spacing
-14 Consistent column spacing

Various other guidelines are also violated in this bad table, including those pertaining to identification of multipage displays and display of control options.

Graphics display data sets in pictorial format, to aid perception of temporal, spatial, or other underlying relations.

-1 • Graphic Display for Data Comparison -1

When users must scan and compare related sets of data quickly, display the data items in graphic format, with backup display of raw data available as a user-selected option.

Comment: People cannot readily assimilate and compare detailed sets of raw data.

Reference: EG 2.2.9.

-2 o Graphic Displays for Monitoring Data Change -2

Provide graphic displays, rather than alphanumeric, when a user must monitor changing data and identify values outside the normal range.

Comment: It is preferable, of course, to program the computer to handle data monitoring, where that is feasible, and to signal detected abnormalities to the user's attention.

Reference: Hanson, Payne, Shiveley and Kantowitz, 1981; Tullis, 1981.

-3 • Conventional Flowchart Orientation -3

When a flowchart displays related data in a logical sequence, the directional orientation of the flowchart should be consistent with familiar reading patterns, i.e., from left to right and from top to bottom for users accustomed to reading English.

Reference: Krohn, 1983.

See also: 2.3-13.

-8 o Consistent Case in Alphabetic Coding -8

When using alphabetic codes, all letters shall either be upper case or else lower case.

Comment: For data display, upper case labels will be somewhat more legible. For data entry, computer logic should not distinguish between upper and lower case codes, because the user will find it hard to remember any such distinction.

Reference: BB 1.3.3.

See also: 1.0-24.

-9 o Combining Letters and Numbers -9

When codes combine letters and numbers, group the characters of each type together, rather than interspersing numbers and letters.

Example: Letter-letter-number ("HW5") will be read and remembered more accurately than letter-number-letter ("H5W").

Comment: Unfortunately, there are common instances in which this practice has not been followed, such as the coding of British and Canadian postal zones.

Reference: BB 1.5.1; MS 5.15.3.5.8.

-10 o Short Codes -10

When arbitrary codes must be remembered by the user, they should be no longer than four or five characters.

Exception: When a code is meaningful, such as a mnemonic abbreviation or a word, it can be longer.

Reference: BB 1.5.2; MS 5.15.3.5.8.

See also: 1.0-14.

-5 • Definition of Display Codes -5

When codes are assigned special meaning in a display, provide a definition at the bottom of the display that replicates the code being defined.

Example: The legend on a map is a common example.

Example: For a color code, each definition should be displayed in the appropriate color, e.g., "RED = hostile" in red.

Reference: BB 7.6.1.

See also: 4.4-17.

-6 • Consistent Coding Across Displays -6

Assign to symbols, and other codes as well, consistent meanings from one display to another.

Comment: When coding is not consistent, the user's task of display interpretation may be made more difficult than if no auxiliary coding were used at all.

Reference: BB 3.6.1, 7.6.2; MS 5.15.3.3.1.

See also: 2.0-13, 4.0-13.

-7 • Alphanumeric Coding -7

Consider alphanumeric characters for auxiliary coding in display applications where basic data presentation is not already alphanumeric (e.g., graphics).

Comment: Select alphanumeric codes that are visually distinct for visual displays, and phonetically distinct for auditory displays (or in any application where displayed codes must be spoken).

Reference: EG Table 1.

See also: 1.0-17.

-3 • Meaningful Codes -3

Adopt meaningful or familiar codes, rather than arbitrary codes.

Example: A three-letter mnemonic code (DIR = directory) is easier to remember than a three-digit numeric code.

Comment: An arbitrary code, such as a Social Security Number, may eventually become familiar through frequent use.

Reference: BB 3.6.2; MS 5.15.3.3.1.

See also: 2.4-4.

-4 o Familiar Coding Conventions -4

Adopt display (and data entry) codes that conform with accepted abbreviations and general user expectations.

Example: Use M for "male", F for "female", rather than arbitrary digits 1 and 2. In color coding, use red for danger.

Comment: If in doubt, an interface designer can survey prospective users to determine just what their expectations may be.

See also: 2.4-3, 2.4-29, 4.0-14.

Coding refers to distinctive means for highlighting different categories of displayed data for user attention.

-1 • Highlighting Critical Data -1

Provide distinctive coding to highlight important data items requiring user attention, particularly when those items are displayed infrequently.

Example: Such items might include recently changed data, or discrepant data exceeding acceptable limits, or data failing to meet some other defined criteria.

Comment: Note that "highlight" is used here in its general sense, meaning to emphasize or make prominent, and is not restricted to any particular method of display coding such as brightening or inverse video.

Comment: Position coding might suffice, i.e., displaying important items consistently in a particular location, as when an error message appears in a space otherwise left blank. But auxiliary codes may still be needed to highlight important items, even if they are positioned consistently.

Reference: EG 2.1.3, 2.3.12; MS 5.15.3.3.1.

See also: 4.0-12.

-2 • Coding by Data Category -2

Provide display coding in applications where the user must distinguish rapidly among different categories of displayed items, particularly when those items are distributed in an irregular way on the display.

-15 o Data Grouped by Importance -15

Where some displayed data items are particularly important, i.e., provide significant information and/or require immediate user response, consider grouping those items at the top of the display.

Reference: BB 1.8.2; Tullis, 1981.

-16 o Data Grouped by Frequency -16

Where some data items are used more frequently than others, consider grouping those items at the top of the display.

Comment: Principles of data grouping also apply to the display/listing of control options.

Comment: These principles for data grouping in display formatting are essentially the same as those recommended for display/control layout in equipment design.

Reference: BB 1.8.2.

See also: 3.1.3-19.

-17 o Data Grouped Alphabetically or Chronologically -17

When there is no appropriate logic for grouping data by sequence, function, frequency or importance, adopt some other principle, such as alphabetical or chronological grouping.

Reference: BB 1.8.1.

See also: 2.1.1-18.

-12 o Grouping for Data Comparison -12

If users must analyze sets of data to discern similarities, differences, trends, and relationships, structure the display format so that the data are consistently grouped.

Example:

Cost			Output		
Actual	Predicted	Difference	Actual	Predicted	Difference
947	901	+ 46	83	82	+ 1
721	777	- 56	57	54	+ 3
475	471	+ 4	91	95	- 4

Reference: BB 1.8.6; Tullis, 1981.

-13 o Data Grouped by Sequence of Use -13

Where displayed data are used in some spatial or temporal order, consider grouping those data by sequence of use to preserve that order.

Example: Data in an electronic display should match the order of items in an associated paper data form.

Reference: BB 1.8.2; PR 4.10.7.

See also: 1.4-23, 1.4-25, 2.1.4-3.

-14 o Data Grouped by Function -14

Where sets of data are associated with particular questions or related to particular functions, consider grouping each set together to help illustrate those functional relationships.

Reference: BB 1.8.2; Tullis, 1981.

-9 • Display Title at Top -9

Begin every display with a title or header, describing briefly the contents or purpose of the display; leave at least one blank line between the title and the body of the display.

Reference: BB 1.1.1, Table 1; PR 4.5.2.

See also: 4.2-6.

-10 o Command Entry, Prompts, Messages at Bottom -10

Reserve the last several lines at the bottom of every display for status and error messages, prompts, and command entry.

Comment: Assuming that the display is mounted physically above the keyboard, which is the customary placement, the user can look back and forth from keyboard to display more easily when prompts and command entry area are at the bottom of the display.

Comment: As a corollary to this recommendation, when separate command sets are associated with different display windows, such as options for display control (size of the window, positioning, etc.), those should probably be shown at the bottom of each window.

Reference: PR 4.5.3; Granda, Teitelbaum and Dunlap, 1982.

See also: 3.1.5-2, 4.0-7.

-11 • Logical Data Organization -11

Display formats should group data items on the basis of some logical principle, considering trade-offs derived from task analysis.

Reference: BB 1.8.1.

-5 o Page Labeling -5

In multi-paged displays, label each page to show its relation to the others.

See also: 2.6-5, 2.6-6, 4.2-7.

-6 • Windows for Related Data Sets -6

Where a user might need to view several different kinds of data jointly, allow the user to define and select separate data windows that will share a single display frame.

Comment: Depending upon user needs (and system capability), data windows might appear simultaneously as segments of a joint display, might be overlaid in varying degrees so as to obscure one another, or might be displayed sequentially at the user's option. In the latter condition, multiple display windows will differ little from multiple display pages, except perhaps in speed of sequential access.

-7 o Integrated Display -7

When coherent display is required to aid user perception of relations among data sets, provide those data in an integrated display rather than partitioning them into separate windows.

Reference: EG 2.3.2.

See also: 2.3-4, 2.6-1.

-8 o Adequate Window Size -8

When a display window must be used for data scanning, design the window size to display more than one line.

Reference: Elkerton, Williges, Pittman and Roach, 1982.

-2 o Distinctive Display Elements -2

Make different elements of display formats clearly perceptible to users, and distinctive from one another.

Example: Different display areas, or "windows", can be separated by spacing (where space permits); outlining can also be used to separate different areas, so that displayed data, control options, instructions, etc., are distinct from each other.

Reference: EG 2.3; MS 5.15.3.1.5; Stewart, 1980.

See also: 3.0-8, 4.0-8.

-3 • Paging Crowded Displays -3

When a display output contains too much data for presentation in a single page, partition the data into separately displayable pages.

Comment: And provide convenient control procedures to allow users to move easily from one page to another.

Reference: BB 4.4.1, 4.4.2; Stewart, 1980.

See also: 2.6-6.

-4 o Related Data on Same Page -4

When partitioning displays into multiple pages, take into account the type of data, and display functionally related data items together on one page.

Comment: This recommendation is easily followed for text displays, involving primarily the elimination of "widows", considerate placement of illustrations, etc. For displayed data forms and tables, data grouping and continuation of headers must be considered. The partitioning of graphics displays may require radical redesign.

See also: 2.3-7.

Format refers to the organization of different types of data in a display to aid assimilation of information.

-1 • Consistent Format -1

Adopt a consistent organization for the location of various display features, insofar as possible, for all displays.

Example: One location might be used consistently for a display title, another area might be reserved for data output by the computer, and other areas dedicated to display of control options, instructions, error messages, and user command entry.

Exception: It might be desirable to change display formats in some distinctive way to help a user distinguish one task or activity from another, but the displays of any particular type should still be formatted consistently among themselves.

Comment: The objective is to develop display formats that are consistent with accepted usage and existing user habits. Consistent display formats will help establish and preserve user orientation. There is no fixed display format that is optimum for all data handling applications, since applications will vary in their requirements. However, once a suitable format has been devised, it should be maintained as a pattern to ensure consistent design of other displays.

Reference: BB 1.1, 1.8.5; EG 2.2.5, 2.3, 2.3.3; MS 5.15.3.3.4; Stewart, 1980.

See also: 4.0-6.

-2 o Only Necessary Data Displayed -2

Each display should provide the user only the information needed at any given point in a transaction sequence; do not overload displays with extraneous data.

Example: (Good)

CODE	DATA TYPE
su =	Summary
d =	Detailed list
se =	Sequences

Select data to be displayed: __ __

(Bad)

CODE	DATA TYPE	DATE IMPLEMENTED
su =	Summary	5-17-82
d =	Detailed list	7-14-82
se =	Sequences	9-25-82

Select data to be displayed: __ __

Comment: Extraneous data will prevent or slow user assimilation of needed information. Where user information requirements cannot be accurately determined in advance of interface design, and/or are expected to be variable, provide on-line user options for data selection, display coverage, and suppression.

Reference: BB 1.7, 1.8.10; EG 3.1.4; MS 5.15.4.6.2; Stewart, 1980; Tullis, 1981.

See also: 2.0-2, 4.0-5.

Density refers to how much data must be output in any display; ideally, provide just what is needed and no more.

-1 • Necessary Data Displayed -1

Each display should provide the user all of the information needed at any given point in a transaction sequence.

Example: Header information should be retained, or else generated anew, when paging/scrolling data tables.

Comment: The user should not have to remember data from one display to the next.

Comment: If data display requirements seem to exceed display capacity, or (more probably) the user's ability to assimilate information from the display, an interface designer should consider whether the user's information handling task can effectively be broken into smaller steps. Prototype testing may be required to determine optimum display densities for critical tasks.

Reference: BB 4.3.6; EG 2.3.15; Stewart, 1980.

See also: 2.0-1, 2.0-2, 2.6-1, 4.0-5, 4.4-1.

Some displays must combine text, form, tabular and/or graphic elements to aid information assimilation by their users.

-1 • Mixing Text with Figures -1

When tables and/or graphics are combined with text, place each figure as near as possible to its first citation in the text, preferably on the same display page.

Exception: If a figure is cited at several points in the text, then it might be desirable to permit optional display of the figure at user request, perhaps as a temporary window overlay at each point of citation.

Exception: If a figure is cited at several points in printed text, and particularly if that text may be accessed at different places by its readers (as in the case of these present design guidelines), then it might be desirable to group figures consistently at a particular location, such as at the end of each section.

Comment: People may not bother to find and look at a figure if it is displayed separately from its citation in the text.

Reference: Whalley and Fleming, 1975.

-4 • Standardized Graphics Symbology -4

Determine standard meanings for graphic symbols, and then use them consistently within a system, and among systems having similar operational requirements.

-11 • Special Symbols -11

Consider using special symbols, such as asterisks, arrows, etc., to draw attention to selected items in alphanumeric displays.

See also: 4.3-17.

-12 o Consistent Use of Special Symbols -12

When using special symbols to code alert conditions, use the only for that purpose; do not display them under other conditions.

See also: 2.4-6.

-13 o Markers Close to Items Marked -13

When a special symbol is used to mark a word, separate the symbol from the beginning of the word by a space.

Comment: A symbol immediately adjacent to the beginning of a word will impair legibility.

Reference: Noyes, 1980.

-14 • Shape Coding -14

Consider coding with geometric shapes to help users discriminate different categories of data on graphic displays.

Comment: Approximately 15 different shapes can be distinguished readily. If that "alphabet" is too small, it may be possible to use component shapes in combination, as in some military symbol codes.

Reference: EG Table 1.

-15 o Establishing Standards for Shape Coding -15

When shape coding is used, assign codes based on established standards or conventional meanings.

Example: A number of international, national, and organizational standards for shape coding exist, and those should be followed where they apply.

Comment: Although shape codes can often be mnemonic in form, their interpretation will generally rely on learned association as well as immediate perception. Existing user standards must be taken into account by the display designer.

Reference: MS 5.15.3.3.6.

See also: 2.4-6, 4.0-13.

-16 • Line Coding -16

For graphic displays, consider using auxiliary methods of line coding, including variation in line type (e.g., solid, dashed, dotted) and line width ("boldness").

Comment: Perhaps three or four line types might be readily distinguished, and two or three line widths.

Reference: EG 2.3.

-17 o Underlining for Emphasis -17

When a line is added simply to mark or emphasize a displayed item, place it under the designated item.

Comment: A consistent convention is needed to prevent ambiguity in the coding of vertically arrayed items; underlining is customary, and does not detract from word legibility.

Comment: For words from the Roman alphabet, underlining probably detracts from legibility less than would "overlining".

Reference: MS 5.15.3.3.5.

-18 o Coding by Line Length -16

Consider using codes with lines of varying length for applications involving spatial categorization in a single dimension.

Example: The length of a displayed vector might be used to indicate distance (or speed).

Comment: Perhaps four lengths can be reliably distinguished in practical use. Long lines will add clutter to a display, but may be useful in special applications.

Reference: EG Table 1.

-19 o Coding by Line Direction -19

Consider using codes with lines of varying direction for applications involving spatial categorization in two dimensions.

Example: The angle of a displayed vector might be used to indicate direction, i.e., heading or bearing.

Comment: Users can make fairly accurate estimates of angles for lines displayed at ten-degree intervals.

Reference: Smith, 1962a.

-20 • Size Coding -20

Consider size coding, i.e., varying the size of displayed alphanumerics and other symbols, only for applications where displays are not crowded.

Comment: Perhaps as many as five sizes might be used for data categorization, but two or three will probably prove the practical limit except for printed displays.

Reference: EG Table 1; MS 5.15.3.3.6.

-21 o Adequate Differences in Size -21

For size coding, a larger symbol should be at least 1.5 times the height of the next smaller symbol.

Comment: Increase in symbol height must usually be accompanied by a proportional increase in width to preserve a constant aspect ratio and so facilitate symbol recognition.

Reference: MS 5.15.3.3.6.

-22 • Brightness Coding -22

Consider coding by differences in brightness for applications that only require discrimination between two categories of displayed items; i.e., treat brightness as a two-valued code, bright and dim.

Example: A data form might display dim labels and bright data items, in order to facilitate data scanning.

Comment: Perhaps as many as four brightness levels might be used, but at some risk of reduced legibility for the dimmer items. It will be safer to reserve brightness as a two-valued code, particularly for displays whose overall intensity can be adjusted at the terminal by the user.

Reference: EG 2.1.4, Table 1.

See also: 1.4-14.

-23 o Brightness Inversion -23

When a capability for brightness inversion is available, i.e., where bright characters on a dark background can be changed under computer program control to dark on light or vice versa (so-called "reverse video"), consider brightness inversion for highlighting displayed items that require user attention.

Comment: Brightness inversion is obviously limited to use as a two-valued code, i.e., a displayed item is either shown with standard or inverted brightness. If brightness inversion is used for alerting purposes, as recommended here, it should be reserved consistently for that purpose, and not be used for general highlighting.

Reference: PR 3.3.4.

See also: 2.4-6.

-24 • Color Coding -24

Consider color coding for applications where users must distinguish rapidly among several categories of data, particularly when data items are dispersed on the display.

Example: Different colors might be used effectively in a position display to distinguish friendly, unknown, and hostile aircraft tracks, or alternatively to distinguish among aircraft in different altitude zones.

Comment: Color is a good auxiliary code, where a multicolor display capability is available. A color code can be overlaid directly on alphanumerics and other symbols without significantly obscuring them. Color coding permits rapid scanning and perception of patterns and relationships among dispersed data items.

Comment: Perhaps as many as 11 different colors might be reliably distinguished, or even more for trained observers; but as a practical matter it will prove safer to use no more than five or six.

Reference: BB 7.2; EG Table 1; MS 5.15.3.3.7; Smith, 1963a; Smith and Thomas, 1964; Smith, Farquhar and Thomas, 1965.

-25 o Conservative Use of Color -25

Employ color coding conservatively, using relatively few colors and only to designate critical categories of displayed data.

Comment: Arbitrary use of many colors may cause displays to appear "busy" or cluttered, and may reduce the likelihood that significat color coding on other displays will be interpreted appropriately and quickly by the user.

Reference: BB 7.1.

-26 o Adding Color to Formatted Displays -26

Add color coding after displays have already been designed as effectively as possible in a monochrome format.

Comment: Do not use color coding in an attempt to compensate for poor display format; redesign the display instead.

Reference: BB 7.3.

-27 o Redundant Color Coding -27

When color coding is used, make it redundant with some other display feature, such as symbology; do not code only by color.

Comment: Displayed data should provide necessary information even when viewed at a monochromatic display terminal, or hard-copy printout, or when viewed by a user with defective color vision.

Reference: BB 7.4, 7.6.3; MS 5.15.3.3.7.

-28 o Unique Assignment of Color Codes -28

When color coding is used, each color should represent only one category of displayed data.

Comment: Color will prove the dominant coding dimension on a display. If several different types of data are displayed in red, say, they will have an unwanted visual coherence that may hinder proper assimilation of information by the user.

Reference: BB 7.6.1; Smith and Thomas, 1964.

-29 o Conventional Assignment of Color Codes -29

Choose colors for coding based on conventional associations with particular colors.

Example: In a display of accounting data, negative numbers might be shown as red, corresponding to the customary use of red ink for that purpose.

Example: Red is associated with danger (in our society), and is an appropriate color for alarm conditions. Yellow is associated with caution, and might be used for alerting messages or to denote changed data. Green is associated with normal "go ahead" conditions, and might be used for routine data display. White is a color with neutral association, which might be used for general data display purposes.

Comment: Other associations can be learned by the user if color coding is applied consistently.

Reference: BB 7.7.1, 7.7.2, 7.7.3; MS 5.15.4.6.1.f.

See also: 2.4-4, 4.0-13, 4.0-14, 4.3-17.

-30 o Limited Use of Blue -30

Choose the color blue only for background features in a display, and not for critical data.

Example: Blue might be used in shading background areas in graphic displays, where its lower apparent brightness could possibly be of benefit.

Comment: The human eye is not equally sensitive to all colors, nor are its optics color-corrected. Blue symbols appear dimmer than others, and are more difficult to focus.

Reference: BB 7.6, 7.7.5.

-31 • Blink Coding -31

Consider blink coding for applications where a displayed item implies an urgent need for user attention.

Comment: Blinking symbols are effective, if used sparingly, in calling the user's attention to displayed items of unusual significance. Blinking characters may have somewhat reduced legibility, and may cause visual fatigue if used too much.

Comment: Perhaps three or four blink rates might be reliably distinguished, but it will probably prove safer to use blinking as a two-level code, i.e, blinking versus non-blinking.

Reference: BB 1.10.2, 1.10.3; EG Table 1; MS 5.15.3.3.2; Smith and Goodwin, 1971b; Smith and Goodwin, 1972.

See also: 2.4-32, 4.3-17.

-32 o Blinking Marker Symbols -32

When a user must read an item that is blink coded, consider adding an extra symbol such as an asterisk to mark the item, and then blinking that marker symbol rather than blinking the item itself.

Comment: This practice will draw attention to an item without detracting from its legibility.

Reference: BB 1.10.3; Smith and Goodwin, 1971b.

See also: 2.4-31.

-33 o Optimal Blink Rate -33

When blink coding is used, the blink rate should be 2 - 5 Hz, with a minimum duty cycle (ON interval) of 50 percent.

Comment: Although equal ON and OFF intervals are often specified, an effective code can probably be provided even when the OFF interval is considerably shorter than the ON (perhaps a wink, rather than a blink), as in occulting lights used for Navy signaling.

Reference: BB 1.10.4; MS 5.15.3.3.2.

-34 • Coding with Texture, Focus, Motion -34

Consider other visual coding dimensions for special display applications, including variation in texture, focus, and motion.

Comment: Texture can be useful for area coding in graphic displays. Only two levels of focus are feasible, clear and blurred, with the risk that blurred items will be illegible. Perhaps 2 - 10 degrees of motion might be distinguished, in display applications where motion is an appropriate and feasible means of coding.

Reference: EG 2.3.

-35 • Auditory Coding -35

Consider auditory displays as a means of supplementing visual display, or as an alternative means of data output in applications where visual displays are not feasible.

Example: Auditory signals may be helpful in alerting users to critical changes in a visual display.

Example: Auditory output might be used to permit telephone access to computer-stored data.

Exception: Auditory display may be impractical in situations where a high ambiant noise level prevents accurate listening.

Comment: As compared with visual displays, an auditory display offers a potential advantage in attracting a user's attention; a user does not have to "listen at" an auditory display in order to hear it. On the other hand, auditory displays suffer from a number of comparative disadvantages. Auditory displays generally do not offer as great a range of coding options. Auditory displays do not permit easy scanning to discern critical data items, or items that may have been missed at first listening. For human listeners with normal vision, auditory displays do not provide a natural representation of spatial relations.

Reference: MS 5.15.3.9.1.

See also: 1.3-31.

-36 o Distinctive Auditory Coding -36

For auditory displays, employ distinctive sounds to code items requiring special user attention.

Example: A variety of signals might be available, including sirens, bells, chimes, buzzers, and tones of different frequency.

Comment: Tones may be presented in sequence to enlarge the signal repertoire.

Reference: Smith and Goodwin, 1970.

See also: 4.3-17.

-37 o Voice Coding -37

For auditory displays with voice output, consider using different voices to distinguish different categories of data.

Comment: At least two voices, male and female, could be readily distinguished, and perhaps more depending upon fidelity of auditory output, and listening conditions.

Reference: Smith and Goodwin, 1970.

Display generation refers to the means for specification of data outputs, either by a user or automatically.

-1 • User Selection of Data for Display -1

For general information handling applications, allow users to specify the data for displayed output, particularly for frequently used displays.

See also: 2.0-2, 2.0-7, 2.9-1.

-2 • Display Identification Labels -2

When a user participates in selection of data for display, assign to each display an identifying label, i.e., an alphanumeric code or abbreviation that can facilitate display requests by the user.

Reference: BB 1.2.3, MS 5.15.3.1.13.

See also: 4.2-6.

-3 o Meaningful Display Labels -3

The display identification label should be unique, short, but meaningful enough to be remembered easily.

Comment: As conceived here, the display label serves as a shorthand identification. The label does not take the place of a full, descriptive title. The full title would be displayed separately.

Comment: Where flexibility is desired, it may be good practice to let a user assign names to the particular sets of data that constitute commonly used displays, either as formal names or else as nicknames associated by the computer with the formal names.

See also: 2.3-9, 4.2-6.

-4 o Consistent Format for Display Labels -4

Place the identifying label used for display selection in a prominent and consistent location on each display.

Example: The top left corner of the display might be used for this purpose.

Reference: BB 1.2.4.

See also: 2.3-1, 4.0-6, 4.2-6.

-5 • Fast Response to Display Request -5

System response to simple requests for data display should take no more than 0.5 to 1.0 second.

Comment: When display output is paced in segments (blocks, paragraphs, etc.), response time refers to output of the first segment. The recommended response time of 0.5 to 1.0 second should apply when a user makes a request (usually perceived as "simple") for the next page of a multi-page display, or when a display begins to move in response to a scrolling command. Responses to requests for new displays may take somewhat longer, perhaps 2 to 10 seconds, particularly if the user perceives such a request to involve more complicated operations, such as accessing different files, transforming data, etc.

Comment: The desirability of rapid display generation is often discounted in practice, particularly for general purpose, time-shared systems where other practical design considerations may dictate slower computer response. For dedicated systems, however, those designed to help perform defined information handling tasks, rapid response should be an explicit design goal, with computer output capabilities sized accordingly.

Reference: EG Table 2.

See also: 3.0-11, 4.2-2, 4.2-3.

-6 o Signaling Completion of Display Output -6

If display generation is slow, and/or if a series of displays are generated automatically, notify the user when display output is complete.

Example: A non-obtrusive auditory signal such as a chime should suffice for this purpose.

-7 • Regenerating Changed Data -7

Where the computer must regenerate a display to update changed data, consider regenerating only those changed items if that will speed display output.

Comment: The critical design issue here is speed of display. It may be easier for a programmer to regenerate an entire display than to change just one item. But if that results in slower computer response to the user, then the added work of programming selective display regeneration may be worthwhile.

-8 o Replacing Changed Data -8

Where the computer must regenerate a display to update changed data, erase old data items before adding new data items to the display.

Comment: The objective here is to avoid any momentary user confusion that might result from watching portions of old data being overwritten and partially overlapped by portions of new data.

Providing context, consistency, and flexibility is important in sequence control as well as in other aspects of user interface design. Several guidelines proposed here deal explicitly with the need to define and maintain context for users.

With regard to consistency of sequence control, it should be emphasized that users of information systems regard their computer as a tool necessary to perform a job. As a tool, they expect the computer to perform efficiently, reliably, and predictably. They will not regard the computer as an intriguingly unpredictable toy with which to play games. Elements of surprise that might be entertaining in a game will be frustrating in a tool.

Neither will users want to regard their computer as a puzzle to be solved, a challenging device whose intricacies promise pleasure in mastery. Where a programmer might be intrigued by the problems of instructing a computer to perform a difficult task, an ordinary user of the system may merely be irritated by the complexity of a computer tool. Where smart shortcuts are provided to perform particular tasks in particular ways, the ordinary system user may resent the extra learning involved, and the extra memory load, rather than appreciate the elegance of saving keystrokes.

This argument for consistent control rather than smart shortcuts has been made elsewhere (e.g., Reisner, 1981) but merits continual repetition. Perhaps the most frequent mistake made by designers of user interface software is to provide smart shortcuts instead of consistent control procedures. In every instance, the designer's intent is to help users -- by shortening a particular command, by saving a logically redundant keystroke, or by making sequence control more efficient for a knowledgeable user with perfect memory. But no real users fit that description. Real users depend upon consistent interface design to set practical limits on what they must learn and remember about their computer tools.

In accord with this argument, many of the guidelines proposed here deal in some way with the need to provide consistent logic for sequence control. A consistent interface design -- where actions are all taken in the same way, where displayed control options are all formatted in the same way, where error messages are all worded in the same way, and so on -- may seem dull to its designers. It may even seem dull to some of its users. But it should prove easy to learn. Smart shortcuts, i.e., the special design features that can make particular control actions more efficient, should be provided only as optional extras, not needed by novice users but offering some flexibility for experienced users.

The selection of dialogue types based on anticipated task requirements and user skills seems straightforward, at least for simple cases. Computer-initiated question-and-answer dialogues are suited to routine data entry tasks, where data items are known and their ordering can be constrained; this type of dialogue provides explicit prompting for unskilled, occasional users. Form-filling dialogues permit somewhat greater flexibility in data entry, but may require user training. When data entries must be made in arbitrary order, perhaps mixed with queries as in making airline reservations, then some mixture of function keys and coded command language will be required for effective operation, implying a moderate to high level of user training.

One important aspect of dialogue choice is that different types of dialogue imply differences in system response time for effective operation. In a repetitive form-filling dialogue, for example, users may accept relatively slow computer processing of a completed form. If the computer should take several seconds to respond, a user probably can take that time to set one data sheet aside and prepare another. But several seconds delay in a menu selection dialogue may prove intolerable, especially where the user must make an extended sequence of selections in order to complete an action.

To categorize these differences, an estimate of the implied requirement for user training and for system response time is given below for eight dialogue types. Cumulative experience and specific requirements of a particular task may modify such estimates.

Dialogue Type	Required User Training	Tolerable Speed of System Response
Question and Answer	Little/None	Moderate
Form Filling	Moderate/Little	Slow
Menu Selection	Little/None	Very Fast
Function Keys	Moderate (potentially little)	Very Fast
Command Language	High	Moderate/Slow
Query Language	High/Moderate	Moderate
Natural Language	Moderate (potentially little)	Fast
Interactive Graphics	High	Very Fast

SECTION 3

SEQUENCE CONTROL

Sequence control refers to user actions and/or computer logic that initiate, interrupt, or terminate transactions. Sequence control governs the transitions from one transaction to the next. General design objectives include consistency of control actions, minimized need for control actions, minimized memory load on the user, with flexibility of sequence control to adapt to different user needs. Techniques of sequence control require explicit attention in user interface design, and many published guidelines deal with this topic.

The importance of good design for controlling user interaction with a computer system is emphasized by Brown, Brown, Burkleo, Mangelsdorf, Olsen and Perkins:

> One of the critical determinants of user satisfaction and acceptance of a computer system is the extent to which the user feels in control of an interactive session. If users cannot control the direction and pace of the interaction sequence, they are likely to feel frustrated, intimidated, or threatened by the computer system. Their productivity may suffer, or they may avoid using the system at all.
>
> (1983, page 4-1)

Complete user control of the interaction sequence and its pacing is not always possible, of course, particularly in applications using computer aids for monitoring and process control. The actions of an air traffic controller, for example, are necessarily paced in some degree by the job to be done. As a general principle, however, it is the user who should decide what needs doing and when to do it.

A fundamental decision in user interface design is selection of the dialogue type(s) that will be used to implement sequence control. Here "dialogue" refers to the sequence of transactions that mediate user-system interaction. Interface design will often involve a mixture of two or more dialogue types, since different dialogues are appropriate to different jobs and different kinds of users. Recognition of appropriate dialogue types at the outset of system development will facilitate the design of user interface software and help ensure the effectiveness of future system operation.

Changes to the software design of data display functions may be needed to meet changing operational requirements.

-1 • Flexible Design for Data Display -1

When data display requirements may change, which is often the case, provide some means for users (or a system administrator) to make necessary changes to display functions.

Comment: Display characteristics that may need to be changed include those represented in these guidelines, namely, the types of data that are displayed, the selection and aggregation of displayed data, the formats for display partitioning, plus changes in display density, coding, coverage, update, and suppression logic.

Comment: Many of the preceding guidelines in this section imply a need for design flexibility. Much of that needed flexibility can be provided in initial interface design. Some guidelines, however, suggest a possible need for subsequent design change, and those guidlines are cited below.

See also: 2.0-2, 2.0-7, 2.0-10, 2.2-2, 2.3-6, 2.5-1, 2.5-3

Suppression refers to user control of display coverage by temporary deletion of specified data categories.

-1 • Temporary Suppression of Displayed Data -1

When standard data displays are used for special purposes, allow users to temporarily suppress the display of data not needed for the current task.

Comment: Data selections made originally for one purpose may not be appropriate for another. When task requirements shift rapidly, it may be more efficient to suppress temporarily the display of unneeded data categories, rather than to re-generate a display with different selection criteria.

See also: 2.0-1.

-2 • Labeling Display Suppression -2

When data have been suppressed from a display, annotate the display with some appropriate label to remind users that data have been suppressed.

-3 • Signaling Changes to Suppressed Data -3

When data have been suppressed from a display, warn users if some significant (but not displayed) change is detected in the computer processing of new data.

-4 • Resuming Display of Suppressed Data -4

When data have been suppressed from a display, provide users with some means to quickly restore the display to its complete, originally selected form.

Comment: In some applications, it may be desirable to restore suppressed data automatically, after expiration of a predetermined time-out, rather than relying on the user to remember to do it.

-5 o Labeling Display Freeze -5

When an updated display is frozen, annotate that display with some appropriate label to remind users of that status.

Reference: MS 5.15.3.4.4.

See also: 4.4-10.

-6 o Signaling Changes to Frozen Data -6

When a display being updated in real time is frozen, warn users if some significant (but not displayed) change is detected in the computer processing of new data.

See also: 4.4-10.

-7 o Resuming Update after Display Freeze -7

When a display being updated in real time has been frozen, and then a user elects to resume update, the computer should resume display update at the current real-time point, unless otherwise specified by the user.

Comment: In some applications, a user might wish to resume display update at the point of stoppage, and so display change would thenceforth lag real-time data change. Or perhaps a user might choose to see a speeded "replay" of interim changes to regain current display status. As a general practice, however, such options risk confusion.

Reference: MS 5.15.3.4.3.

-3 o Visual Integration of Changing Graphics -3

If users must visually integrate changing patterns on a graphic display, update the data at a rate appropriate to human perceptual abilities for the kind of data change to be recognized.

Example: Effective display of radar data for track detection and monitoring requires repeated, sequential output of stored data frames, and the timing of successive frames is critical for optimizing pattern perception.

Comment: Slowly developing patterns may be seen more easily with time compression, i.e., with rapid display of sequentially stored data frames. Fast changing data may require time expansion, i.e., slowed output, to aid pattern perception. For critical applications, experimentation with prototype displays will often be required to determine proper timing. In applications where the timing of display update is variable, it may help to note the currently selected time scale on the display.

Comment: Similar considerations apply to auditory displays, where speeding or slowing sound signals may aid pattern recognition.

Reference: MS 5.15.3.6.3.

-4 • Display Freeze -4

When a display is automatically updated, allow users to stop the process ("freeze", "stop action") at any point, in order to examine changed data more deliberately, where operational requirements permit.

Comment: For some applications, it may also prove helpful if the user can step incrementally forward or back in the time sequence, frame by frame.

Comment: Display freeze may help in information handling tasks involving careful analysis of changing data. But display freeze will often not be feasible in monitoring tasks where a user must deal with sequential data changes as they occur.

Reference: MS 5.15.3.4.3.

Display update refers to regeneration of changed data to show current status, by user request or automatically.

-1 • Automatic Display Update -1

When displayed data are changing as a result of external (non-user) action, users should be able to request automatic update (computer regeneration) of changed data, and be able to control the update rate, where operational requirements permit.

Reference: MS 5.15.3.4.2.

-2 • Readability of Changing Data -2

If users must accurately read changing data values, ensure that updated data are displayed long enough to be read.

Comment: A current design standard specifies that for accurate reading, data should be displayed in a fixed position and updated no more than once per second. In some applications, however, a slower update rate may be required. When in doubt, test user performance with prototype displays to determine appropriate update rates.

Comment: If users need only to monitor general trends in changing data values, and do not need to take exact readings, somewhat faster update rates may be acceptable. Again, prototype testing will sometimes be needed to determine appropriate update rates.

Reference: MS 5.15.3.4.1.

-10 o Labeling Windowing Functions -10

When a windowing orientation is maintained consistently, choose names for display framing functions that refer to movement of the display frame (or window) and not to movement of the displayed data.

Example: The command "Up 10" should mean that the display frame will move up ten lines, with the effect that ten lines of previous data will appear at the top of the display, and ten lines of later data will disappear at the bottom.

-11 o Labeling Scrolling Functions -11

When a scrolling orientation is maintained consistently, choose names for display framing functions that refer to movement of the data being displayed, and not to movement of the display frame (or window).

Example: The command "Up 10" should mean that displayed data will move up ten lines behind the (conceptually fixed) display frame, with the effect that ten lines of previous data will disappear from the top of the display, and ten lines of new data will appear at the bottom.

Reference: EG 2.3.16.

-8 o Windowing with Free Cursor Movement -8

In applications where users can mover a cursor freely within a page of displayed data, adopt windowing rather than scrolling as the conceptual basis of display framing.

Example: Full-screen editing is a common application involving free cursor movement.

Comment: Since displayed data will be perceived as fixed during cursor movement, considerations of joint compatibility suggest that displayed data remain conceptually fixed during movement of the display frame or window. Indeed, it may be possible to use the same arrow-labeled function keys to control both cursor movement and windowing.

Reference: Morrill and Davies, 1961.

See also: 1.3-3.

-9 • Labeling Display Framing Functions -9

When users will access different systems with different conventions for windowing/scrolling, interface designers should always refer to display framing in functional terms (e.g., "forward" and "back", or "next" and "previous"), and should avoid wording that implies spatial orientation (e.g., "up" and "down"), for user instructions, key labels, etc.

Comment: Control of display framing functions might be implemented by keys marked with arrows, avoiding verbal labels altogether.

Comment: Note that "forward" and "back" are potentially ambiguous because of the contradictory use of those words in referring to movement within books.

-6 o Numbering Display Pages -6

When display output is more than one page, annotate each page to indicate display continuation.

Example: The phrase "page x of y" is commonly used for this purpose.

Comment: When a display extends over just a few pages, and when a user is not expected to care about any particular page, then it may be sufficient to identify the pages "first", "continued", and "last" rather than assigning them numbers.

Comment: A recommended format is to identify pages by a note immediately to the right of the display title. With such a consistent location, the page note might be displayed in dimmer characters. Leading zeros should not be used in the display of page numbers.

Reference: MS 5.15.3.1.12; PR 4.5.5, 4.10.4.

See also: 2.3-3, 4.2-7.

-7 • Consistent Orientation for Display Framing -7

Adopt a consistent orientation for display framing throughout interface design, so that users can either 1) conceive data as moving behind a fixed display frame, commonly called "scrolling", or 2) conceive the display frame as a window moving over a fixed array of data, here called "windowing".

Comment: Ideally a consistent orientation for display framing would be maintained across all systems. Certainly that orientation should be consistent within any one system.

Comment: A user can adapt to either concept, if it is maintained consistently. Both concepts have some precedent in natural experience. Moving a specimen beneath the fixed eyepiece of a microscope illustrates scrolling. Moving a telescope across a fixed scene illustrates windowing. Tests seem to indicate that windowing is the more natural concept for inexperienced users, causing fewer errors, and hence is the preferred option when other considerations are equal.

Reference: Bury, Boyle, Evey and Neal, 1982.

-4 o Labels for Multipage Tables -4

For a large table that exceeds the capacity of one display frame, ensure that users can see column headings and row labels in all displayed sections of the table.

Reference: BB 1.9.6; MS 5.15.3.5.4.

See also: 1.5-1.

-5 • Annotating Display of Continued Data -5

When lists or tables are of variable length, and may extend beyond the limits of a single display page, inform users when data are continued on another page and when data are concluded on the present page.

Example: Incomplete lists might be marked "continued on next page", or "continued", or "more". Concluding lists might add a note "end of list", or "end".

Exception: Short lists whose conclusion is evident from the display format need not be annotated in this way.

Reference: BB 1.9.7.

See also: 4.2-7.

Framing refers to user control of data coverage by display movement, including paging, scrolling, offset, etc.

-1 • Integrated Display -1

Whenever possible, include all data relevant to a user's current transaction in one display page (or "frame").

Comment: This recommendation is particularly important when critical data items must be compared for effective information assimilation by a user. Do not rely on a user to remember data accurately from one display frame to another.

Reference: EG 3.4.4.

See also: 2.0-1, 2.2-1, 2.3-4, 2.3-7, 4.0-5, 4.4-1.

-2 • Easy Paging -2

When requested data exceed the capacity of a single display frame, give users some easy means to move back and forth over displayed material by paging (or windowing/scrolling).

Example: Dedicated function keys might be provided for paging forward and back.

Comment: Note that critical data requiring integrated display for effective assimilation should be included in a single frame, and not dispersed over several pages. Paging is acceptable when the user is looking for a specific data item, but not when the user must discern some relationship among different sets of data.

Reference: BB 4.4.1, 4.4.2, 4.4.9; EG 6.3.8; MS 5.15.3.1.11.

-3 o Continuous Numbering in Multipage Lists -3

When a list of numbered items exceeds one display page, number the items continuously in relation to the first item on the first page.

Reference: EG 2.3.10.

-9 • Printing Displays Locally -9

When displayed data are of potential long-term interest, give users some easy means to print paper copies locally, within security restraints.

Comment: Users should not have to remember displayed data. Optional printout permits a user to record data from one display to compare with another, and so deal with situations where the system designer has not anticipated the need for such comparison.

Comment: Users should not have to take notes or transcribe displayed data manually. That practice under-utilizes the data handling potential of the computer, and risks transcription errors by the user.

Reference: BB 4.4.6; EG 4.2.14; MS 5.15.9.2; PR 4.10.1.

See also: 1.3-27, 6.2-7, 6.4-5.

Ideal flexibility would permit experienced users to undertake whatever task or transaction is needed, at any time. Although this may not always prove feasible, the interface designer should try to provide the maximum possible user control of the on-line transaction sequence. As a simple example, a user who is scanning a multi-page data display should be able to go either forward or back at will. If interface software only permits stepping forward, so that users must cycle through the entire display set to reach a previous page, that design is inefficient. Users should also be able to interrupt display scanning at any point to initiate some other transaction. Such simple flexibility is relatively easy for the designer to achieve, and indeed is commonly provided.

More difficult are transactions that involve potential change to stored data. Here again users will need flexibility in sequence control, perhaps wishing to back up in a data entry sequence to change previous items, or to cancel and restart the sequence, or to abort the sequence altogether and escape to some other task. The interface designer can provide such flexibility through use of suspense files and other special programmed features. This flexibility requires extra effort from the software designer and programmer. But that extra effort is made only once, and is a worthwhile investment on behalf of future users who may interact with their computer system for months or even years.

Of course, flexibility of sequence control has pitfalls. Just as users can make mistakes in data entry, so also will users make mistakes in sequence control. The interface designer must try to anticipate user errors and ensure that potentially damaging actions are difficult to take. In most data entry tasks, for example, simple keying of data items should not in itself initiate computer processing. The user should have to take some further, explicit action to ENTER the data. The interface logic should be designed to protect the user from the consequences of inadvertently destructive actions. Any large-scale erasure or deletion of data, for example, should require some sort of explicit user confirmation, being accomplished as a two-step process rather than by a single keystroke. (This provides a software analogy to the physical barriers sometimes used to protect critical hardware controls from accidental activation.) Some well-designed systems go a step further and permit the user to reverse (UNDO) a mistaken action already taken.

One form of flexibility frequently recommended is the provision of alternate modes of sequence control for experienced and inexperienced users. In a command-language dialogue, optional guidance might be provided to prompt a beginner step by step in the composition of commands, whereas an experienced user might enter a

complete command as a single complex input. Some such flexibility in the user interface is surely desirable -- so that the computer can interpret halting, stepwise control inputs, as well as fluent, coherent commands.

More generally, however, it may be desirable to include redundant modes of sequence control in user interface design, perhaps involving combinations of different dialogue types. As an example, menu selection might be incorporated to provide easy sequence control for beginners, but every display frame might also be formatted to include a standard field where an experienced user could enter complete commands more efficiently. Examples of this approach have been provided by Palme (1979).

Another way to provide flexibility in sequence control is through specific tailoring of display formats. Consider, for example, a menu selection dialogue in which sequence control is exercised through lightpen selection among displayed control options. For any particular display frame it might be possible to display just three or four options most likely to be selected by a user at that point in the task sequence, plus a general purpose OPTIONS selection that could be used to call out a display of other (less likely) commands. Thus, on the first page of a two-page display set, one of the likely commands would be NEXT PAGE; but on the second page that command would be replaced by its more likely complement, PREV PAGE.

This approach illustrates two design ideas. The first comes close to being a general rule for sequence control: make the user's most frequent transactions the easiest to accomplish. The second idea is the reliance on context to improve flexibility. These general ideas concerning sequence control are reflected in the specific design guidelines proposed in the following pages.

- Sequence control refers to user actions and/or computer logic that initiate, interrupt, or terminate transactions.
- The user-computer dialogue for sequence control will be of different types, to meet task requirements and user needs.

- Transaction selection refers to control actions and computer logic that initiate transactions.
- For flexibility of sequence control, a user should be able to interrupt and redirect ongoing transactions.
- Designers should ensure that current context which affects the outcome of control actions is evident to users.
- Users will make errors, and interface design must facilitate the detection and correction of those errors.
- Users may need to control the logic and operation of alarms and alerting signals.
- Changes to the software design of control functions may be needed to meet changing operational requirements.

Objectives:

Consistency of control actions
Minimal control actions by user
Minimal memory load on user
Compatibility with task requirements
Flexibility of sequence control

Sequence control refers to user actions and/or computer logic that initiate, interrupt, or terminate transactions.

-1 • Flexible Sequence Control -1

Provide flexible means of sequence control so that users can accomplish necessary transactions involving data entry, processing, retrieval and transmission, or can obtain guidance as needed in connection with any transaction.

Example: In scanning a multipage display the user should be able to go forward or back at will. If user interface design permits only forward steps, so that the user must cycle through the entire display series to reach a previous page, that design is deficient.

Comment: Necessary transactions should be defined in task analysis prior to software design.

Reference: PR 4.0.

-2 • Minimal User Actions -2

Control entries should be simple, particularly for real-time tasks requiring fast user response; control logic should permit completion of a transaction sequence with the minimum number of actions consistent with user abilities.

Example: A user should be able to print a display without having to take a series of other actions first, such as calling for the display to be filed, specifying a file name, then calling for a print of that named file.

Example: For long, multipage displays, it should be possible to request a particular page directly, without having to take repetitive NEXT PAGE or PREVIOUS PAGE actions.

Exception: A destructive action will be less likely to be taken by mistake, if it is designed to be different or distinctive, requiring extra user actions.

Comment: Shortcuts via direct commands should allow experienced users to by-pass intervening steps that may help beginners. The computer should be programmed to handle automatically any intervening processing that may be required, informing the user what has been done if that becomes necessary (as in the case of a detected error).

Reference: BB 2.4.1, 4.5; MS 5.15.4.6.4.

See also: 3.0-3, 4.0-22.

-3 • Control Matched to User Skill -3

Ensure that the means of sequence control are compatible with user skills, permitting simple step-by-step actions by beginners, but permitting more complex command entry by experienced users.

Comment: This will generally require a user interface that provides a mix of dialogue types, i.e., command entry augmented by other techniques such as function keys and menu selection.

Reference: BB 4.5; Gilfoil, 1982.

See also: 4.4-26, and Section 3.1.

-4 • User Initiative in Sequence Control -4

Software design should permit initiative and control by the user; try to anticipate user requirements and provide appropriate user control options and computer responses in all cases.

Comment: In most applications, users should be able to interrupt or terminate transactions once they have been initiated (see Section 3.3). Users will sometimes change their minds and decide that an initiated transaction is not what was wanted after all.

Comment: Software logic should be "bulletproofed" to anticipate every possible action by a user, no matter how improbable, providing an appropriate computer response to random (or even malicious) inputs as well as correct entries and likely errors. In particular, a dialogue should never reach a dead end with no further action available to the user. If a user makes an entry inappropriate to current processing logic, the computer should simply display an advisory message that the input cannot be recognized and indicate the available options as to what can be done next.

Reference: BB 4.2.5.1; PR 2.2.

See also: 4.4-5.

-5 • Control by Explicit User Action -5

Permit users to control transaction sequencing by explicit action; defer computer processing until an explicit user action has been taken.

Example: When a user is keying an extended data entry, the computer should not interrupt the user to require immediate correction of any entry error, but instead should wait for the user's ENTER action.

Example: When a user is composing a command to accomplish some transaction, the computer should not interrupt the user by responding as soon as it recognizes a partial entry, but instead should wait for the user's ENTER action.

Exception: In automated process control applications, emergency conditions may take precedence over current user transactions, and a computer-generated warning might interrupt user actions.

Exception: In routine, repetitive data entry transactions, successful completion of one entry may lead automatically to initiation of the next, as in keying ZIP codes at an automated post office.

Comment: If the computer interrupts a user, it pre-empts the initiative in sequence control, in effect forcing the user into an error correction (or some other) sequence conceived by the interface designer, and not necessarily a sequence that would be chosen by the user.

Comment: Some interface designers devise computer interruptions that they suppose will help a user, as when they program a computer to complete a partial command automatically as soon as it recognizes the user's intended entry. Many users, however, will find unexpected computer interruptions more disconcerting than helpful, and may become confused at least momentarily as to just what they had intended.

Comment: In general, computer detection of problems with current user entries can be negotiated at the conclusion of a transaction, before it is implemented. Nondisruptive alarms or advisory messages can be displayed to report computer monitoring of external events so that the user can choose when to deal with them.

See also: 1.0-8, 1.1-4, 1.4-1, 4.0-2, 6.0-3, 6.3-8.

-6 • Consistent User Actions -6

Sequence control actions should be consistent in form and consequences; employ similar means to accomplish ends that are similar, from one transaction to the next, from one task to another, throughout the user interface.

Comment: In particular, there should be some standard, consistent routine for a user to initiate and terminate the transaction sequences that comprise different tasks. Do not require users to learn different command names to terminate different tasks, or remember to terminate one task by command and another by function key.

Reference: Reisner, 1981.

See also: 4.0-1.

-7 • Logical Transaction Sequences -7

When designing a sequence of related transactions for some information handling task, employ task analysis to ensure that those transactions will constitute a logical unit or subtask from the users' viewpoint, and to determine what control options to provide users at any point.

Comment: A logical unit to the user is not necessarily the same as a logical unit of the computer software that mediates the transaction sequence. It might be, for example, that a user should enter ten items of data in a single transaction, because those data all come from one particular paper form, even though the computer will use five of those items for one purpose and five items for another in its subsequent internal processing.

Reference: PR 5.1; Stewart, 1980.

See also: 4.0-1.

-8 • Distinctive Display of Control Information -8

Design all displays so that features relevant to sequence control are distinctive in position and/or format.

Comment: Relevant features include displayed options, command entry areas, prompts, advisory messages, and other displayed items (titles, time signals, etc.) whose changes signal the results of control entries.

See also: 2.3-2, 4.0-6.

-9 • Displayed Context -9

If the consequences of a given control entry will differ depending upon context established by a prior action, then some appropriate means of context identification should always be displayed to the user.

Example: If activating a DELETE key establishes a mode, so that subsequent selection of a PAGE key will erase a page of data rather than simply advancing to display the next page, then some indication of that established DELETE mode should be displayed to the user.

Comment: Do not rely on the user always to remember prior actions, nor to understand their current implications.

See also: 4.4-10, and Section 3.4.

-10 • Consistent Terminology for Sequence Control -10

For instructional material, display labeling, on-line guidance and other messages to users, adopt consistent terminology to refer to sequence control.

Example: Various words and phrases might be used, such as "control input", "command entry", "instruction", "request", "function call", etc. The practice adopted in these guidelines is to call general sequence control actions "control entry". More specific terminology is sometimes used here, such as "command entry" for keyed control entries composed by the user, "code entry" for keyed selections from displayed menus, etc.

See also: 4.0-15.

-11 • Feedback for Control Entries -11

The computer should acknowledge every control entry immediately; for every action by the user there should be some apparent reaction from the computer, either by

(1) execution of a requested transaction if that produces immediately apparent results, or

(2) a message indicating completion of the transaction, or

(3) a message indicating that execution is in progress or deferred, or

(4) a message that the control entry requires correction or confirmation.

Example:

(1) A user requests NEXT PAGE, and the next page is displayed as an apparent result of the transaction.

(2) A user requests printout at a remote facility, and the computer displays a confirming message, AIRFIELD FILE HAS BEEN SENT TO PRINTER.

(3) A user enters data, and the computer displays an interim message to indicate processing of the entries, AIRFIELD FILE IS BEING UPDATED.

(4) A user requests file display, and the computer displays an error message, "AIRFELD" FILE NOT RECOGNIZED.

Comment: In particular, the absence of computer response is not an acceptable means of indicating that a control entry is being processed.

Comment: "Immediately" as used in this guideline must be interpreted in relation to the response time requirements of different dialogue types.

Reference: BB 4.3.1, 4.3.2; EG 4.2.5; MS 5.15.5.2, 5.15.5.3, 5.15.2.1.3.

See also: 1.0-2, 1.0-11, 1.0-12, 3.0-12, 4.2-1, 4.2-3, and Section 3.1.

-12 o Indicating Completion of Processing -12

When processing in response to a control entry is lengthy, give the user some positive indication of subsequent completion, and appropriate related information.

Comment: If a user is currently involved in some new transaction, then completion of processing for a prior transaction should be indicated by nondisruptive display of an appropriate advisory message.

Comment: If the outcome of a completed transaction may imply the need for further user action, that should be indicated to the user.

Reference: BB 4.3.1; MS 5.15.5.2.

See also: 3.0-11, 4.2-4.

-13 o Compatibility with User Expectations -13

Ensure that the results of any control entry are compatible with user expectations, so that changes in the state or value of controlled elements are displayed in an expected or natural form.

Example: A control entry of NEXT PAGE should show the next frame of a current display, and should not jump off to some other internally defined "page" in the computer's data base.

Example: When data or command entry is accomplished by function key, that key should be labeled ENTER (or some functionally equivalent word) and should result in entry and computer acknowledgment of the data or appropriate response to the command.

Comment: Compatibility between user action and system response is an important concept in human engineering design. Interface designers should not assume that user expectations will match their own. User expectations can be discovered by interview, questionnaire, and/or prototype testing. Where no strong user expectations exist with respect to a particular design feature, then designers can help establish valid user expectations by careful consistency in interface design.

Reference: MS 5.15.4.1.12; Smith, 1981b.

See also: 1.0-9, 1.1-15, 1.1-19, 3.0-6, 4.2-1.

-14 • User-Paced Sequence Control -14

Permit users to pace control entries, rather than requiring users to keep pace with computer processing or external events.

Comment: User pacing will let control entries be made in accord with a user's current needs, attention span, and time available.

Comment: When user-paced control does not seem feasible, as in critical process control applications, reconsider the general approach to task allocation and user interface design, perhaps providing greater system automation to ensure timely response.

See also: 1.0-7.

-15 • Appropriate Computer Response Time -15

Ensure that the speed of computer response to user control entries is appropriate to the transaction involved; in general, the response should be faster for those transactions perceived by a user to be simple.

Example: Computer response to a likely control entry, such as NEXT PAGE, should be within 0.5-1.0 second; response to other simple entries should be within 2.0 seconds; error messages should be displayed within 2-4 seconds.

Comment: Interface designers may need to consult with the intended system users to decide upon appropriate computer response times for different transactions.

Reference: EG Tables 2-3; MS Table XXIX; Miller, 1968.

See also: 1.0-3, 3.0-16, 4.2-2, 4.3-11.

-16 • Control Availability -16

A user should be able to make control entries as needed; and a sequence of control entries should not be delayed or paced by delays in computer response.

Comment: It is recommended that control delays or lockouts not exceed 0.2 seconds. In some applications, however, longer delay may be tolerable, particularly if that has the effect of reducing variability in computer response time.

See also: 1.0-3, 3.0-15.

-17 o Indicating Control Lockout -17

If control entries must be delayed pending computer processing of prior entries, then indicate that delay to the user.

Example: If processing delay results in control lockout, that could be signaled by disappearance of the cursor from the display, or perhaps by a notable change in the shape of the cursor, accompanied by an auditory signal.

Comment: In some applications it may be desirable to ensure that the keyboard and other control devices are automatically locked until the user can begin a new transaction. This would be true when processing the current transaction will affect the results of subsequent user actions. In other applications, it may be possible to permit users to continue work while previous transactions are still being processed.

Comment: Deletion or change of a displayed cursor may not be in itself a sufficient indicator of keyboard lockout. Auditory signals will be particularly helpful to skilled touch typists, who may not look at the display when transcribing data entries.

Comment: Following control lockout, computer readiness to accept further entries should be indicated to the user.

See also: 3.0-18, 4.1-4.

-18 o Interrupt to End Control Lockout -18

In situations where control lockout does occur, provide the user with an auxiliary means of control entry, such as a special function key, to abort a transaction causing extended lockout.

Comment: Such an interrupt capability will be especially helpful if a user recognizes that an error has been made and wants to stop an unneeded transaction, acting like an UNDO command.

Comment: Alternatively, for some transactions it may be helpful to design this interrupt as an END command that stops ongoing processing without canceling it. For example, if a user has asked the computer to scroll ahead in a long file display, that user may simply wish to stop at a certain point rather than returning to the beginning.

See also: 3.0-17, 3.3-6.

-19 • Control by Simultaneous Users -19

When several users must interact with the system simultaneously, ensure that control entries by one user do not interfere with those of another.

Comment: This requires careful user interface design for applications where joint, coordinated actions must be made by a group of users.

Reference: MS 5.15.4.6.5.

See also: 6.5-2.

The user-computer dialogue for sequence control will be of different types, to meet task requirements and user needs.

-1 • Dialogue Matched to User and Task -1

Consider task requirements and associated user characteristics when choosing dialogue type(s) and designing sequence control dialogue.

Example: Where untrained users must choose among a fixed set of options, as in automated bank teller machines, then labeled function keys will probably suffice for sequence control. Where options may be chosen from a larger set, as in public information systems, then menu selection will prove a more efficient dialogue type.

Example: In a task where users must make data and control entries in an arbitrary order, perhaps mixed with queries (as in making flight reservations when talking with a customer), then some mixture of function keys and coded command entries will be required for effective operation.

Comment: A simple dictum here is, "Know the user." However, if user characteristics are variable, which is usually the case, then provide a variety of dialogue types based on analysis of task requirements.

Reference: MS 5.15.4.1.8; Martin, 1973.

See also: 3.0-3, 4.4-26.

-2 • Appropriate Computer Response Time -2

Ensure that the speed of computer response to user entries is appropriate to the type of dialogue; in general, the response to menu selections, function keys, and most entries during graphic interaction should be immediate.

Comment: It is generally thought that maximum acceptable delay for computer response to menu selection by lightpen is 1.0 second; for key activation is 0.1 second; for cursor positioning by lightpen (as in graphic line drawing) 0.1 second.

Comment: If computer response time will be slow, consider choosing other dialogue types, such as command entry.

Reference: EG Tables 2-3; MS Table XXIX; Miller, 1968.

See also: 3.0-15, 4.2-2.

In a type of dialogue that is simple for novice users, the computer displays questions for the user to answer.

-1 • Question-and-Answer Dialogue -1

Consider question-and-answer dialogue primarily for routine data entry tasks, where data items are known and their ordering can be constrained, where users will have little or no training, and where computer response is expected to be moderately fast.

Example: In the automated collection of medical history data, a computer might follow contingent branching logic in posing questions for patients to answer.

Comment: Brief question-and-answer sequences can be used to supplement other dialogue types for special purposes, such as for log-on routines, or for resolving ambiguous control or data entries.

Comment: Where computer response to any single user entry may be slow, then the aggregate time required to process a series of questions and answers may be very slow. In such a case, consider form filling as an alternative dialogue type, where the user can enter a set of related "answers" as a single transaction.

-2 • Questions Displayed Singly -2

In question-and-answer dialogues, display each question separately; do not require users to answer several questions at once.

Comment: A user may become confused in trying to deal with several questions at once, particularly if the number of questions is variable from one transaction to another.

-3 • Recapitulating Prior Answers -3

When a series of computer-posed questions are interrelated, display answers to previous questions when those will provide context to help a user answer the current question.

Comment: Do not rely on a user to remember prior answers.

Comment: An alternative way to request a related series of user entries is to adopt a form-filling dialogue rather than question-and-answer.

See also: 3.4-2.

-4 • Sequence Compatible with Source Documents -4

When questions prompt entry of data from a source document, the question sequence should match the data sequence in the source document.

See also: 1.4-23.

A computer may display a form of labeled fields to aid composition and review of data and/or control entries.

-1 • Form Filling for Data Entry -1

Consider form filling for tasks where some flexibility in data entry is needed, such as the inclusion of optional as well as required items, where users will have moderate training, and/or where computer response may be slow.

Example: Form filling might be an appropriate dialogue type for a computer system that helped users calculate income tax obligations.

Comment: Specific recommendations for the design of form-filling dialogues are presented in Section 1.4 for data entry and in Section 2.1.2 for data display.

Reference: MS 5.15.4.3.1.

See also: Section 1.4, Section 2.1.2.

-2 • Form Filling for Control Entry -2

Consider form filling as an aid for composing complex control entries, i.e., as a supplementary means of prompting command or query entries.

Example: For a complex data retrieval request, a displayed form might indicate the various control parameters that could be specified.

-3 • Defaults for Control Entry -3

Consider form filling as a means of displaying default values for the parameters in complex control entries.

Comment: Defined defaults permit users to enter potentially complicated commands/queries by relatively simple actions. If such defaults have been defined, they should be indicated to users. A displayed form permits users to review (and confirm or change) default control values, just as they might review displayed defaults for data entry.

Comment: When only a few control parameters are involved, in relatively simple commands/queries, then it may be possible simply to prompt users with guidance messages, rather than by displaying a control form.

Reference: EG 4.2.4.

See also: 3.1.5-4.

-4 • Consistent Format for Control Forms -4

Forms for control entry should be consistent in format; their design should generally conform to guidelines for the design of data entry forms.

See also: Section 1.4, Section 2.1.2.

Displayed menus permit control entries by pointing, or by keying option codes, or by associated multifunction keys.

-1 • Menu Selection -1

Consider menu selection for tasks that involve choice among a relatively constrained set of alternative actions, that require little entry of arbitrary data, where users may have relatively little training, and where computer response is relatively fast.

Example: Displayed menus are commonly used to permit selection of text processing functions, for specifying functions in graphic interaction, and for a multitude of other applications.

Comment: Lengthy menus are commonly formatted as separate displays. Task-specific menus, however, can often be incorporated effectively along with data displays, to provide a short list of appropriate control options.

Comment: Menu selection is, of course, a generally good means for control entry by untrained users. Menus can be used in conjunction with other dialogue types, depending upon task requirements. Menu selections might be clarified by brief question-and-answer dialogue.

Comment: Menu selections might be optionally by-passed by command entry, for experienced users. When display output is slow, as for a printing terminal, or for an electronic display constrained by a low-bandwidth channel, it may be tiresome for a user to wait for display of menu options, especially if selections must be made from a sequence of sequentially displayed menus.

Reference: MS 5.15.4.2.1.

-2 • Single Selection per Menu -2

Each menu display should permit only one selection by the user.

Comment: Novice users will be confused by any more complicated procedure, such as a "Chinese menu" requiring one choice from Column A, two from Column B, etc.

Reference: PR 4.6.5.

-3 • Single-Column List Format -3

When multiple menu options are displayed in a list, display each option on a new line, i.e., format the list as a single column.

Exception: Listing in multiple columns may be considered where shortage of display space dictates a compact format; if there are only a few options, those might be displayed in a single row.

Comment: A single-column list format will aid scanning and assimilation of available options, especially for novice users.

Reference: MS 5.15.4.2.7.

See also: 2.1.1-17.

-4 • Menu Selection by Pointing -4

When menu selection is the primary means of sequence control, and especially if choices must be made from extensive lists of displayed control options, then permit selection by direct pointing (e.g., by touch display, lightpen, etc.).

Comment: Pointing directly at a displayed option guarantees good display-control compatibility. Users do not have to note associated option codes and enter them by key actions.

Reference: MS 5.15.2.5.1, 5.15.4.2.2; Thompson, 1971.

See also: 1.1-12.

-5 o Large Pointing Area for Option Selection -5

If menu selection is to be accomplished by pointing, as on touch displays, design the acceptable area for pointing to be as large as consistently possible, including at least the area of the displayed option label plus a half-character distance around that label.

Comment: The larger the effective target area, the easier the pointing action will be, and the less risk of error in selecting a wrong option by mistake.

Reference: BB 2.12; EG 2.3.13, 6.1.3.

See also: 1.1-13.

-6 o Dual Activation for Pointing -6

If menu selection is to be accomplished by pointing, provide for dual activation, in which the first action designates (positions a cursor at) the selected option, followed by a separate second action that makes an explicit control entry.

Example: On a touch display, the computer might display a separate ENTER box that can be touched by a user to indicate that the cursor has been properly positioned.

Comment: The two actions of cursor placement and entering should be compatible in their design implementation. If the cursor is positioned by keying, then an ENTER key should be used to signal control entry. If the cursor is positioned by lightpen, provide a dual-action "trigger" on the lightpen for cursor positioning and control entry.

Comment: This recommendation for dual activation of pointing assumes that accuracy in selection of control entries is more important than speed. In some applications that may not be true. Interface design will involve a trade-off considering the criticality of wrong entries, ease of recovery from wrong entries, and user convenience in making selections.

Reference: Foley and Wallace, 1974.

See also: 1.0-8, 1.1-4, 3.0-5, 6.0-3.

-7 • Menu Selection by Keyed Entry -7

When menu selection is a secondary (occasional) means of control entry, and/or only short option lists are needed, then consider accomplishing selection by keyed entry.

Comment: Options might be selected by entering associated codes which are included in the displayed menu listing. Alternatively, if menu labels can be listed near a display margin, then options might be selected by pressing adjacent multifunction keys.

-8 o Standard Area for Code Entry -8

When menu selection is accomplished by code entry, provide a standard command entry area (window) where users enter the selected code; place that entry area in a fixed location on all displays.

Comment: In a customary terminal configuration, where the display is located above the keyboard, command entry should be at the bottom of the display, in order to minimize user head/eye movement between the display and the keyboard.

Comment: Experienced users might key coded menu selections in a standard area identified only by its consistent location and use. If the system is designed primarily for novice users, however, that entry area should be given an appropriate label, such as ENTER CHOICE HERE: ___.

Reference: MS 5.15.4.2.2; PR 4.6.3.

See also: 3.1.5-2, 4.0-6.

-9 • Feedback for Menu Selection -9

When a user has selected and entered a control option from a menu, if there is no immediately observable natural response then the computer should display some other acknowledgment of that entry.

Comment: An explicit message might be provided. In some applications, however, it may suffice simply to highlight the selected option label (e.g., by brightening or inverse video) when that would provide an unambiguous acknowledgment.

Reference: MS 5.15.4.1.12.

See also: 1.1-5, 3.0-11, 4.2-1, 4.2-10.

-34 o Stacking Menu Selections -34

For menu selection by code entry, when a series of selections can be anticipated before the menus are displayed, permit a user to combine those selections into a single "stacked" entry.

Comment: If necessary, stacked sequential entries might be separated by some character, such as a space, slash, comma or semicolon. It would be preferable, however, if they were simply strung together without special punctuation. Computer interpretation of an unpunctuated string will require letter codes (by preference) or fixed-digit number codes for option selection.

Reference: BB 2.9.

See also: 3.1.3-12.

-32 o Return to General Menu -32

When hierarchic menus are used, permit users to take only one simple key action to return to the general menu at the top level.

Example: [See sample displays at the end of this section.]

Comment: This action could be considered analogous to the RESTART option proposed as an interrupt for sequence control.

See also: 3.1.3-24, 3.3-5.

-33 • By-Passing Menu Selection with Command Entry -33

Permit experienced users to by-pass a series of menu selections and make an equivalent command entry directly.

Comment: In effect, a command entry might specify an option anywhere in a hierarchic menu structure, permitting a user to jump down several levels, or to move directly from one branch to another.

Comment: If a command by-passes only a portion of the complete menu sequence, and so does not yet specify a control entry, then display the appropriate next menu to guide completion of the control entry.

Reference: BB 2.8, 4.5; PR 4.7.3.

See also: 3 0-2, 3.1.3-11, 4.4-26.

-29 o Control Options Distinct from Menu Branching -29

Format the display of hierarchic menus so options that actually accomplish control entries can be distinguished from options that merely branch to other menu frames.

Example: [See sample displays at the end of this section.]

Comment: In some applications, it may prove efficient to design "hybrid" menus which display one branch of the menu hierarchy elaborated to include all of its control options while other branches are simply indicated by summary labels. In such a hybrid menu, it will help orient users if options that accomplish control actions are highlighted in some way to distinguish them from options that will result in display of other frames of the hierarchic menu.

-30 o Consistent Design of Hierarchic Menus -30

When hierarchic menus are used, ensure that display format and selection logic are consistent at every level.

Reference: MS 5.15.4.1.6.

See also: 4.0-6.

-31 o Return to Higher-Level Menus -31

When hierarchic menus are used, permit users to take only one simple key action to return to the next higher level.

Comment: This action could be considered analogous to the BACKUP option proposed as an interrupt for sequence control.

Reference: BB 4.4.4.

See also: 3.3-4.

-27 o Automatic Cursor Placement -27

On dedicated menu displays (i.e., for menus not included with data displays), when menu selection is by pointing the computer should place the cursor automatically at the first listed option; when menu selection is by code entry, place the cursor in the command entry area.

Comment: When menu selection is by code entry, for some applications it may increase the efficiency of sequence control if a null entry is recognized as a default to the first displayed option (assuming that the first option is the most likely choice). If that is done, it should be done consistently.

See also: 1.4-26.

-28 • Indicating Current Position in Menu Structure -28

When hierarchic menus are used, display to users some indication of current position in the menu structure.

Example: [See sample displays at the end of this section.]

Comment: One possible approach would be to recapitulate prior (higher) menu selections on the display. If routine display of path information seems to clutter menu formats, then a map of the menu structure might be provided at user request as a HELP display.

Reference: MS 5.15.4.1.6; Billingsley, 1982.

See also: 4.4-4.

-25 o Minimal Steps in Sequential Menu Selection -25

When users must step through a sequence of menus to make a selection, design the hierarchic menu structure to minimize the number of steps required.

Example: [See sample displays at the end of this section.]

Comment: This represents a trade-off against the need for logical grouping in hierarchic menus. Minimize the number of hierarchic levels, but not at the expense of display crowding.

Reference: MS 5.15.4.1.6; Miller, 1981; Snowberry, Parkinson, and Sisson, 1983.

-26 o Easy Selection of Important Options -26

When hierarchic menus are used, design their structure to permit immediate user access to critical or frequently selected options.

Example: [See sample displays at the end of this section.]

Comment: It may be desirable in general purpose systems whose use is varied and unpredictable, to permit users to tailor menu design (particularly the general menu) to their individual needs, so that the options used most frequently will appear first for each user.

Comment: In designing fixed hierarchic menus, if frequent or critical options do appear logically at lower levels, and so will be less accessible, some design alternatives should be considered. For a critical action, some sort of "panic" option might be included in every menu display, or might be implemented by function key. For frequent actions, some special menu display might be provided as a supplementary shortcut to the designed menu hierarchy.

Reference: MS 5.15.4.2.8.

See also: 3.1.4-2.

-23 • Hierarchic Menus for Sequential Selection -23

When menu selection must be made from a long list, and not all options can be displayed at once, provide a hierarchic sequence of menu selections rather than one long multipage menu.

Example: [See sample displays at the end of this section.]

Exception: Where a long list is already structured for other purposes, such as a list of customers, a parts inventory, a file directory, etc., it might be reasonable to require the user to scan multiple display pages to find a particular item. Even in such cases, however, an imposed structure for sequential access may prove more efficient, as when a user can make preliminary letter choices to access a long alphabetic list.

Comment: Beginning users may learn faster and understand better a menu permitting a single choice from all available options, when those can be displayed on one page. However, a single long menu that extends for more than one page will hinder learning and use. The interface designer can usually devise some means of logical segmentation to permit several sequential selections among few alternatives instead of a single difficult selection among many.

Reference: MS 5.15.4.2.6; Dray, Ogden and Vestewig, 1981.

See also: 4.4-4.

-24 o General Menu -24

Provide a general menu of basic options as the top level in a hierarchic menu structure, a "home base" to which a user can always return as a consistent starting point for control entries.

Comment: Return to the general menu might be accomplished by an OPTIONS function key, or by an explicitly labeled option on every display, or by a generally available implicit option.

See also: 3.1.3-32, 3.2-2.

-21 o Logical Ordering of Grouped Options -21

If menu options are grouped in logical subunits, display those groups in a logical order; if no logical structure is apparent, then display the groups in the order of their expected frequency of use.

Example: [See sample displays at the end of this section.]

Reference: PR 4.6.6.

-22 o Labeling Grouped Options -22

If menu options are grouped in logical subunits, give each group a descriptive label that is distinctive in format from the option labels themselves.

Example: [See sample displays at the end of this section.]

Comment: Although this practice might sometimes seem to waste display space, it will help provide user guidance; moreover, careful selection of group labels may serve to reduce the number of words needed for individual option labels.

Reference: MS 5.15.3.1.10.

See also: 4.4-4.

-19 • Logical Ordering of Menu Options -19

List displayed menu options in a logical order; if no logical structure is apparent, then display the options in order of their expected frequency of use, with the most frequent listed first.

Example: (Good) i = Initiate track
m = Move track
d = Delete track

(Bad) d = Delete track
i = Initiate track
m = Move track

Example: [See sample displays at the end of this section.]

Reference: BB 2.9.4; EG 2.3.1; MS 5.15.4.2.5; PR 4.6.6; Palme, 1979.

See also: 2.1.3-6, 2.3-16.

-20 • Logical Grouping of Menu Options -20

Format a menu to indicate logically related groups of options, rather than as an undifferentiated string of alternatives.

Example: In vertical listing of options, subordinate categories might be indented.

Example: [See sample displays at the end of this section.]

Comment: Logical grouping of menu options will help users learn system capabilities.

Comment: When logical grouping requires a trade-off against expected frequency of use, interface designers should resolve that trade-off consistently for those functions throughout the menu structure.

Reference: EG 2.2.8, 2.3; Foley and Wallace, 1974; Liebelt, McDonald, Stone and Karat, 1982; McDonald, Stone and Liebelt, 1983.

See also: 4.4-3.

-17 • Consistent Display of Menu Options -17

When menus are provided in different displays, design them so that option lists are consistent in wording and ordering.

Example: If + PRINT is the last option in one menu, the same print option should not be worded + COPY at the beginning of another menu.

Reference: MS 5.15.4.2.4.

See also: 3.1.3-13, 4.0-15.

-18 • Menus Distinct from Other Displayed Information -18

If menu options are included in a display that is intended also for data review and/or data entry, which is often a practical design approach, ensure that they are distinct from other displayed information; locate menu options consistently in the display and incorporate some consistent distinguishing feature to indicate their special function.

Example: All control options might be displayed beginning with a special symbol, such as a plus sign (+ NEXT, + BACK, etc.)

See also: 2.1.3-9, 4.0-8.

-15 o Complete Display of Menu Options -15

A menu should display all options appropriate at any step in a transaction sequence.

Exception: A familiar set of general control options, i.e., options that are always implicitly available, may be omitted from individual displays, and accessed as needed, perhaps by requesting display of a supplementary menu, or perhaps by function key or command entry.

See also: 4.4-1, and Section 3.2.

-16 o Menu Options Dependent on Context -16

A menu should display only options that are actually available in the current context for a particular user.

Example: Privileged users might be shown more options than regular users.

Example: Displayed file directories should contain only those files actually available to the particular user.

Example: Offer a CHANGE option only to users who are authorized to make changes to the particular data being displayed.

Exception: Menu displays for a system under development might display future options not yet implemented, but such options should be specially marked in some way so that users will understand that they are not available.

Comment: If a user selects a displayed option, and is then told that option is not actually available, an undesirable element of unpredictability has been introduced into the interface design. Users may become uncertain and confused about sequence control. Also irritated.

Reference: BB 1.8.11; MS 5.15.4.2.3.

See also: 3.2-10, 4.4-1.

-13 o Consistent Coding of Menu Options -13

If letter codes are used for menu selection, use them consistently in designating options from one transaction to another.

Example: The same action should not be given different names and hence different codes (F = FORWARD and N = NEXT).

Example: The same code should not be given to different actions (Q = QUIT and Q = QUEUE).

Comment: Different codes for the same action will tend to confuse users and impede learning. The same code for different actions will tend to induce user errors, particularly if those actions are frequently taken. However, this practice may be tolerable when selections are seldom taken and then always from labeled alternatives.

See also: 3.1.3-17, 4.0-15.

-14 • Explicit Option Display -14

When control entries for any particular transaction will be selected from a small set of options, show those options in a menu added to the working display, rather than requiring users to remember them or to access a separate menu display.

Comment: A complete display of control options will sometimes leave little room for display of data. If an extensive menu must be added to a working data display, provide that menu as a separate window that can temporarily overlay displayed data at user request, but can then be omitted again by further user action.

Reference: MS 5.15.4.1.5.

See also: 4.4-5.

-12 • Letter Codes for Menu Selection -12

If menu selections must be made by keyed codes, design each code to be the initial letter or letters of the displayed option label, rather than assigning arbitrary letter or number codes.

Example: (Good) m = Male
f = Female

(Bad) 1 = Male
2 = Female

Exception: Options might be numbered when a logical order or sequence is implied.

Exception: When menu selection is from a long list, line numbers in the list might be an acceptable alternative to letter codes.

Comment: Several significant advantages can be cited for mnemonic letter codes. Letters are easier than numbers for touch typists to key. It is easier to memorize meaningful names than numbers, and so letter codes can facilitate a potential transition from menu selection to command language when those two dialogue types are used together. When menus have to be redesigned, which sometimes happens, lettered options can be reordered without changing codes, whereas numbered options might have to be changed and so confuse users who have already learned the previous numbering.

Comment: USI designers should not create unnatural option labels just to ensure that the initial letter of each will be different. There must be some natural differences among option names, and special two- or three-letter codes can probably be devised as needed to emphasize those differences. In this regard, there is probably no harm in mixing single-letter codes with special multi-letter codes in one menu.

Reference: BB 1.3.6; MS 5.15.4.2.11; Palme, 1979.

[But note contrary advocacy of number codes in BB 1.9.3; EG 2.2.7; PR 4.6.2.]

See also: 4.0-13.

-11 o Option Wording Consistent with Command Language -11

If menu selection is used in conjunction with or as an alternative to command language, design the wording and syntactic organization of displayed menu options to correspond consistently to defined elements and structure of the command language.

Comment: Where appropriate, display cumulative sequences of menu selections in a command entry area until the user signals entry of a completely composed command.

Comment: This practice will speed the transition for a novice user, relying initially on sequential menu selection, to become an experienced user composing coherent commands without such aid.

Reference: MS 5.15.4.2.9.

See also: 3.1.3-33, 4.0-15.

-10 • Menu Options Worded as Commands -10

The wording of menu options should consistently represent commands to the computer, rather than questions to the user.

Example: For option selection by pointing,

(Good) + PRINT

(Bad) PRINT?

[Here it is assumed that "+" has been chosen as the symbol used consistently to distinguish a selectable control option from other displayed items.]

Example: For option selection by code entry,

(Good) p = Print

(Bad) Print? (Y/N)

Comment: Wording options as commands will permit logical selection by pointing, will facilitate the design of mnemonic codes for keyed entry, and will help users learn commands in systems where commands may be used to by-pass menus.

Comment: Wording options as commands implies (properly) that the initiative in sequence control lies with the user. Wording options as questions implies initiative by the computer.

Reference: PR 4.6.8.

See also: 3.1.3-18, 4.0-21.

(Good) Sample Menu Display (Good)

```
W : WORD PROCESSING MENU          GO= General Options

    D : DOCUMENT MANAGEMENT
        C = Create
            CF= Free format
            CL= Letter
            CM= Memo
            CW= Wide format
        E = Edit
        P = Print
        CO= COpy
        RE= Rename
        DE= DElete
        SP= SPelling Check
        I = Index

    T = Transferring documents
    L = List processing
    S = Status information
    U = User profile


ENTER letter code to select action or another menu.
```

These sample displays represent portions of a large menu of word processing functions. The good menu indicates the current position in a hierarchic menu structure. Different levels in the hierarchic structure are indicated by indentation. This menu offers control actions (bolded) for the most frequently used branch ("document management"), along with options to select other branches in the menu hierarchy. In this conceptual design, selection of another branch would produce a similar menu display, offering control actions within the selected branch, and without offering the control actions shown here for document management.

The bad menu is an alternative design for the same functions. That menu lacks hierarchic structure, and shows no distinction between control actions and options that merely select further menus. Using the bad menu, some frequent actions will require several successive menu selections before they can be taken.

(Bad) Sample Menu Display (Bad)

```
C  =  Create a new document

CW =  Create a new wide document

D  =  Delete a document

E  =  Edit an existing document

F  =  Finished -- Exit

I  =  Index of documents

L  =  List Processing

M  =  More Menu selections

P  =  Print a document

S  =  Spelling Error Detection


Type the letters followed by a RETURN
```

This bad menu display violates in some degree several design guidelines in this section:

3.1.3-19 Logical ordering of menu options
-20 Logical grouping of menu options
-21 Logical ordering of grouped options
-22 Labeling grouped options
-23 Hierarchic menus for sequential selection
-25 Minimal steps in sequential menu selection
-26 Easy selection of important options
-28 Indicate current position in menu structure
-29 Control options distinct from menu branching
-32 Return to general menu

Function keys permit control entries by direct selection of labeled keys, rather than from displayed menus.

-1 • Function Keys for Critical Control Entries -1

Consider function keys for tasks requiring only a limited number of control entries, or for use in conjunction with other dialogue types as a ready means of accomplishing critical entries that must be made quickly without syntax error.

Reference: BB 4.4; MS 5.15.2.3.1, 5.15.4.4.

-2 o Function Keys for Frequent Control Entries -2

Consider function keys for frequently required control entries.

Example: Commonly used function keys include ENTER, PRINT, NEXT PAGE, PREV PAGE, OPTIONS, etc.

Comment: When frequently used options are always available via function keys, they need not be included in displayed menus.

Reference: BB 4.4; MS 5.15.2.3.1.

See also: 3.1.3-26, and Section 3.2.

-3 o Function Keys for Interim Control Entries -3

Consider function keys for interim control entries, i.e., for control actions taken before the completion of a transaction.

Example: Function keys will aid such interim actions as DITTO, CONFIRM, and requests for PRINT, or HELP, and also interrupts such as BACKUP, CANCEL, etc.

Comment: Interim control refers to an action taken by a user while working with displayed data, e.g., while still keying data entries or changes, etc. Function keys will aid interim control entries partly because those entries may be frequent. More importantly, however, function keys permit those control entries to be made without special cursor positioning, so that they do not interfere with data entry.

-4 • Distinctive Labeling of Function Keys -4

Label function keys informatively to designate the function they perform; make labels sufficiently different from one another to prevent user confusion.

Example: Logging on to a system should not be initiated by a key labeled PANIC. (This example is cited from an actual design.)

Example: Two keys should not be labeled ON and DN.

Reference: BB 4.4.7; MS 5.15.2.3.9, 5.15.3.1.10.c.

See also: 4.0-10.

-5 o Labeling Multifunction Keys -5

If a key is used for more than one function, always indicate to the user which function is currently available.

Comment: If a key is used for just two functions, depending upon defined operational mode, then alternate illuminated labels might be provided on the key to indicate which function is current. In those circumstances, it is preferable that only the currently available function is visible, so that the labels on a group of keys will show what can be done at any point.

Comment: If key function is specific to a particular transaction, provide an appropriate guidance message on the user's display to indicate the current function.

Reference: MS 5.15.2.4.2, 5.15.2.4.4.

See also: 4.4-10.

-6 • Single Keying for Frequent Functions -6

Keys controlling frequently used functions should permit single key action and should not require double (control/shift) keying.

-7 o Single Activation of Function Keys -7

Any key should perform its labeled function with a single activation, and should not change its function with repeated activation.

Exception: On a very compact keypad, where separate keys are not available to accommodate the range of needed functions, it might be acceptable to group logically related functions on a single key, where repeated key activation would extend the range of control action in a consistent way (e.g., DELETE character, word, sentence, or paragraph with repeated key strokes).

Reference: MS 5.15.2.3.7.

See also: 4.0-2.

-8 • Feedback for Function Key Activation -8

When function key activation does not result in any immediately observable natural response, provide users with some other form of computer acknowledgment.

Comment: Temporary illumination of the function key will suffice, if key illumination is not used for other purposes, such as indicating available options. Otherwise an advisory message should be displayed.

Comment: As an interesting variation, user guidance prior to key activation might be provided, where partial depression of a double-contact function key would explain its use, either by voice output ("talking keyboard") or by visual display of a HELP message.

Reference: MS 5.15.2.3.8; Geiser, Schumacher and Berger, 1982.

See also: 3.0-11, 4.2-1.

-9 • Indicating Active Function Keys -9

When some function keys are active and some are not, indicate the current subset of active keys in some noticeable way, perhaps by brighter illumination.

Comment: This practice will speed user selection of function keys.

Reference: MS 5.15.2.4.3; Hollingsworth and Dray, 1981.

See also: 4.4-10.

-10 o Disabling Unneeded Function Keys -10

When function keys are not needed for any current transaction, temporarily disable those keys under computer control; do not require users to apply mechanical overlays for this purpose.

Comment: If a user selects a function key that is invalid for the current transaction, no action should result except display of an advisory message indicating what functions are available at that point.

Reference: MS 5.15.9.1; PR 4.12.4.5.

See also: 3.2-10, 3.5-1, 6.5-7.

-11 • Single Key for Continuous Functions -11

When a function is continuously available, assign that function to a single key.

Reference: MS 5.15.2.3.4.

-12 • Consistent Assignment of Function Keys -12

If a function is assigned to a particular key in one transaction, assign that function to the same key in other transactions.

Example: A SAVE key should perform the same function at any point in a transaction sequence.

Comment: This becomes a design issue, of course, only in applications where the set of needed functions does vary somewhat from one transaction to another.

Reference: BB 4.4.8; MS 5.15.2.3.2; Foley and Wallace, 1974.

-13 o Consistent Functions in Different Operational Modes -13

When a function key performs different functions in different operational modes, assign similar (or equivalent) functions to the same key.

Example: A particular key might be used to confirm data changes in one mode, confirm message transmission in another, etc.

Example: As a negative example, a key labeled RESET should not be used to save data in one mode, dump data in another, and signal task completion in a third. (This example is cited from an actual design.)

Reference: Stewart, 1980.

-14 • Return to Base-Level Functions -14

When the functions assigned to a set of keys change as a result of user selection, give the user an easy means to return to the initial, base-level functions.

Example: In cockpit design, where multifunction keys may be used for various purposes such as navigation or weapons control, the pilot should be able to take a single action to restore those keys quickly to their basic flight control functions.

Comment: In effect, multifunction keys can provide hierarchic levels of options much like menu selection dialogues, with the same need for rapid return to the highest-level menu.

Comment: For some applications, it may be desirable to automate the return to base-level assignment of multifunction keys, to occur immediately on completion of a transaction and/or by time-out following a period of user inaction. The optimum period for any automatic time-out would have to be determined empirically for each application.

Reference: Aretz and Kopala, 1981.

-15 • Distinctive Location -15

Group function keys in distinctive locations on the keyboard to facilitate their learning and use; place frequently used function keys in the most convenient locations.

Reference: MS 5.15.2.3.6.

See also: 4.0-8.

-16 o Layout Compatible with Use -16

The layout of function keys should be compatible with their importance; give keys for emergency functions a prominent position and distinctive coding (e.g., size and/or color); provide physical protection for keys with potentially disruptive consequences.

See also: 6.5-8.

A command language lets users specify desired control actions by composing messages to a computer.

-1 • Command Language -1

Consider command language dialogue for tasks involving a wide range of control entries, where users may be highly trained and will use the system frequently.

Comment: Command language should also be considered for tasks where control entries may be mixed with data entries in arbitrary sequence, such as when making flight reservations. Such applications will generally require extensive user training.

Reference: MS 5.15.4.5.1; Martin, 1973.

-2 • Standard Display Area for Command Entry -2

When command language is used for sequence control, provide a command entry area in a consistent location on every display, preferably at the bottom.

Comment: Adjacent to the command entry area there should be a defined display window used for prompting entries, for recapitulation of command sequences (with scrolling to permit extended review), and to mediate question-and-answer dialogue sequences (i.e., prompts and responses to prompts).

Reference: EG 2.3; MS 5.15.4.5.7; Granda, Teitelbaum and Dunlap, 1982.

See also: 2.3-10, 3.1.3-8, 4.0-6.

-3 • Functional Command Language Design -3

Design a command language so that users can enter commands in terms of functions desired, without concern for internal computer data processing, storage and retrieval mechanisms.

Example: Users should be able to request display of a data file by name alone, without any further specification such as that file's location in computer storage.

Comment: Where file names are not unique identifiers, the computer should be programmed to determine whatever further context is necessary for identification. Or perhaps the computer should ask the user to designate a "directory" defining the subset of files of current interest.

Reference: MS 5.15.4.1.10.

-4 • Layered Command Language -4

Design a command language so that its features are organized in groups (or "layers") for ease in learning and use.

Example: A user should be able to display the next of a set of received messages with some simple command such as READ NEXT, although a complete command to retrieve any message might include potential specification of which message, from which message list, in which format, to which output device.

Comment: The fundamental layer of the language should be the easiest, allowing use of the system by people with little training and/or limited needs. Successive layers of the command language can then increase in complexity for users with greater skills. In effect, simple versions of commands can be recognized by defaulting all of the optional parameters.

Comment: Control forms might be used to display default options for complicated commands.

Reference: Reisner, 1977.

See also: 3.1.2-3, 4.4-26.

-5 • Familiar Wording -5

Choose words for a command language that reflect the user's point of view, and correspond to the user's operational language.

Example: To transfer a file, the assigned command should be something like TRANSFER, MOVE, or SEND, and not some jargon term like PIP.

Reference: EG 4.1.1, 4.2.12, 4.2.13; MS 5.15.4.5.2.

See also: 4.0-16, 4.0-17.

-6 o Consistent Wording of Commands -6

Design all words in a command language, and their abbreviations, to be consistent in meaning from one transaction to another, and from one task to another.

Example: Do not use EDIT in one place, MODIFY in another, UPDATE in a third, all referring to the same kind of action.

Example: Choose wording so that commands will be congruent with one another, following natural language patterns; if one command is UP, its complement should be DOWN; other natural complements include RIGHT-LEFT, FORWARD-BACK, IN-OUT, PUSH-PULL, RAISE-LOWER, etc.

Reference: EG 4.2.9, 4.2.13; MS 5.15.4.5.6; Carroll, 1982; Demers, 1981.

See also: 4.0-15, 4.0-17.

-7 o Distinctive Wording of Commands -7

Design words in a command language so that they are distinctive from one another, and emphasize significant differences in function.

Comment: In general, do not give different commands semantically similar names, such as SUM and COUNT, or ERASE and DELETE, or QUIT and EXIT. System design abounds with negative examples of similarly named commands which confuse their users: DISPLAY and VIEW (where one command permits editing displayed material and one does not), COMPOSE and CREATE (where one command sends the composed message to an outbox and one leaves the message on the desk), etc. Even experienced users will make errors with such commands.

Comment: Some researchers deal with this question by recommending the use of specific rather than general command names.

Reference: BB 3.7.5; MS 5.15.4.5.3; Barnard, Hammond, MacLean and Morton, 1982.

-8 • User-Assigned Command Names -8

A command language should provide flexibility, permitting the user to assign personal names to files, frequently used commands, etc.

Comment: Frequently used commands should be easy for a user to enter. Where users differ in the frequency of the commands they use, perhaps the designer should provide for flexibility in command naming. On the other hand, users will not be perfectly consistent in specifying command names, and a carefully designed set of commands might well prove better for some applications.

Comment: For users who must move back and forth between different systems with differently defined command languages, some flexibility in command naming might permit those users to establish their own consistent terminology.

Comment: Before users can be allowed to adopt their own assigned command names, the computer must check those names to prevent duplication.

Comment: A potential risk of increased flexibility is increased confusion, if users forget what names they have specified for commands and data files. The computer should maintain a current index of command and file names for on-line user reference.

Reference: Carroll, 1982.

See also: 3.1.5-11, 3.2-18, 4.4-14, 4.4-15.

-9 • User-Requested Prompts -9

Permit users to request computer-generated prompts as necessary to determine required parameters in a command entry, or to determine available options for an appropriate next command entry.

Example: Using a HELP function key, or perhaps simply keying a question mark in the command entry area, would be a satisfactory method of requesting prompts.

Comment: In some applications it may be desirable to let an inexperienced user simply choose a general "prompt mode" of operation, where any command entry produces automatic prompting of (required or optional) parameters and/or succeeding entry options.

Reference: MS 5.15.4.5.8; Demers, 1981.

See also: 4.4-7, 4.4-9.

-10 o General List of Commands -10

Provide a general list of basic commands, with appropriate command format guidance, that will always be available to serve as a "home base" or consistent starting point for composing command entries.

Comment: Such a general list of commands might provide more comprehensive user guidance than is possible when prompting command entry from a working display.

See also: 3.2-2, 4.4-2.

-7 o Cursor Placement for Keyed Entry of Options -7

When users must select options by keyed entry of a corresponding code, the computer should place the cursor in the command entry area at display generation.

Reference: PR 4.7.1.

See also: 4.4-12.

-8 • Displaying Option Codes -8

When users must select options by code entry, display the code associated with each option in a consistent distinctive manner.

Example: In many applications an equal sign can be used to designate option codes, as N = NEXT PAGE, P = PREV PAGE, etc.

-9 • Task-Oriented Wording for Options -9

Wording of control options should be task-oriented to reflect the user's view of the current transaction.

Example: When assigning aircraft to a mission, the relevant control option should be ASSIGN rather than ENTER.

See also: 4.0-17.

-10 • Only Available Options Offered -10

Offer users only control options that are actually available for the current transaction.

Comment: If certain options are not yet implemented, as during system development, or not available for any other reason, those should be annotated on the display.

See also: 3.1.3-16, 3.1.4-10, 6.5-7.

-4 • Indicating Appropriate Control Options -4

Make available to users a list of the control options that are specifically appropriate for any transaction.

Comment: Transaction-specific options might be listed in the working display if there is space for them. Otherwise, they might be displayed in an overlay window at user request.

Comment: Treat control options that are available for almost any transaction as implicit options, which need not be included in a list of transaction-specific options unless they are particularly appropriate to the current transaction. One convenient way to offer implicit options is via function keys, although some experienced users may prefer to select implicit options by command entry.

See also: 3.1.4-2, 4.4-1, 4.4-6.

-5 o Prompting Control Entries -5

Provide users with whatever information may be needed to guide control entries at any point in a transaction sequence, by incorporating prompts in a display and/or by providing prompts in response to requests for HELP.

Reference: MS 5.15.4.1.4.

See also: 4.4-1.

-6 • Cursor Placement for Pointing at Options -6

When users will select among displayed options by pointing, the computer should place the cursor on the first (most likely) option at display generation.

See also: 3.1.3-27, 4.4-12.

-2 • General List of Control Options -2

Provide a general list of basic control options that will always be available to serve as a "home base" or consistent starting point for control entries.

Comment: Return to this starting point can be accomplished by an OPTIONS function key, or by an explicit control option on every display, or by a generally available implicit option.

Comment: Such a capability may be helpful even when all dialogue is user-initiated. It might be the general menu for a menu selection dialogue, or might be a standard starting point for composing command entries.

Comment: However, a user should not be required to return to a display of general options in order to make a control entry. If a user remembers option codes or commands, ideally those control entries could be made from any point in a transaction sequence.

Reference: BB 4.1; PR 3.3.16.

See also: 3.1.3-24, 3.1.5-10, 4.4-2.

-3 o Organization and Labeling of Listed Options -3

The general options list should show control entry options grouped, labeled and ordered in terms of their logical function, frequency and criticality of use, following the general guidelines for menu design.

See also: 4.4-2, 4.4-3, and Section 3.1.3.

Transaction selection refers to control actions and computer logic that initiate transactions.

-1 • User Control in Transaction Selection -1

Permit users to select transactions; computer processing constraints should not dictate sequence control.

Example: A user who wants to interrupt a current activity should not be required by the computer to complete some long sequence of useless transactions.

Comment: When a logical sequence of transactions can be determined in advance, interface design might encourage and help a user to follow that sequence. Guidance may be desirable though constraint is not.

Reference: PR 4.6.7.

See also: 3.0-1, 3.0-4, 3.0-5, 4.0-2.

Graphic interaction permits users to exercise control by pointing or drawing on visual displays.

-1 • Graphic Interaction -1

Consider graphic interaction as a supplement to other forms of user-system dialogue where special task requirements exist.

Comment: Effective implementation of graphic capabilities will require very fast computer response.

Comment: Supplemental graphic depiction of specified logical combinations may help clarify the use of command/query languages.

Reference: Michard, 1982; Shneiderman, 1982.

See also: 3.1.6-7.

-2 • Menu Selection as Complementary Dialogue -2

Consider menu selection by pointing as a means of complementing graphic interaction, permitting compatible selection of both graphic elements and control actions.

Comment: In a graphic interaction dialogue, a user's attention will be concentrated on the display itself, so that pointing at displayed menu options will naturally complement pointing at graphic elements.

Computer processing of natural language could permit a novice user to compose commands without any special training.

-1 o Constrained Natural Language -1

Consider using some limited form of natural language dialogue in applications where task requirements are broad ranging and poorly defined, and where little user training can be provided.

Comment: For untrained users, the seemingly familiar form of a (limited) natural language dialogue may help introduce them to computer capabilities. Such users may manage to do something right from scratch, without having to surmount an initial hurdle of learning more specialized command languages and control procedures. As users gain experience, they may eventually learn more efficient methods of interacting with the system. On the other hand, infrequent computer users may forget whatever training they receive, and so remain novices indefinitely.

Comment: Computer processing of natural language is now being developed on an experimental basis. Current capabilities permit computer recognition of constrained forms of "natural" language, with some limits on vocabulary and syntax. Such constrained natural languages might be considered akin to command languages, with the drawback that they are probably not as carefully designed.

Comment: Do not consider using unconstrained natural language dialogues for current interface design. Even if a computer can be programmed to recognize unconstrained natural language, it is not clear whether that would help in any particular information handling task. A natural language will often cause confusion in communication among its human users. Something better may be needed to mediate human communication with computers. For applications where task requirements are well defined, other types of dialogue will probably prove more efficient.

Reference: Shneiderman, 1981.

-8 • Linking Sequential Queries -8

A query language should be designed to permit the logical linking of sequential queries.

Example: Such linking might be accomplished with referential pronouns ("of them", "of those") that will be recognized by the computer in terms of current context.

-9 • Confirming Large-Scale Retrieval -9

If a query will result in a large-scale data retrieval, require the user to confirm the transaction or else take further action to narrow the query before processing.

Comment: In this regard, it may be helpful to permit a user to set some upper bound for data output, in effect to define what constitutes a "large-scale" retrieval.

Comment: It may help a user to decide whether to confirm or modify a pending query, if the user can request a partial display of the currently specified data output.

-6 • Minimal Need for Quantifiers -6

Query language design should minimize the need for quantifiers in query formulation.

Example: Negative quantifiers ("no", "none", "zero", etc.) are particularly difficult for users to deal with; other potentially confusing quantifiers include indefinite ("some", "any") and interrogative ("how many") forms.

Comment: People have difficulty in using quantifiers unambiguously. When a query language does require quantifiers, it may be helpful to permit a user to select the desired quantifier from a set of sample queries worded so as to maximize their distinctiveness.

-7 • Logic to Link Queries -7

A query language should include logic elements that permit users to link sequential queries as a single entry.

Example: Common links for query formulation include "and", "or", etc.

Comment: However, a query language should be designed so that it does not require logical links. Some logical quantifiers ("greater than", "less than", etc.) may confuse users. As an alternative to logical linking, it may prove helpful to permit a user to formulate a series of simple queries to narrow the set of retrieved data.

Comment: It may help a user to specify logical links accurately if the computer can display a graphical depiction of relations among data sets as those relations are specified during query composition. One researcher recommends Venn diagrams for this purpose.

Reference: Michard, 1982.

See also: 3.1.8-1.

-3 o Coherent Representation of Data Organization -3

Establish one single representation of the data organization for use in query formulation, rather than multiple representations.

Example: If different queries will access different data bases over different routes, a user should not necessarily need to know this.

Comment: Beginners or infrequent users may be confused by different representational models.

Reference: Michard, 1982.

See also: 4.4-14.

-4 • Task-Oriented Wording -4

The wording of a query should simply specify what data are requested, and should not have to tell the computer how to find the data.

Comment: This objective has been called "nonprocedurality", meaning that a user should not have to understand computer procedures for finding data.

Reference: Michard, 1982.

-5 • Flexible Query Formulation -5

Permit users to employ alterative forms when composing queries, corresponding to common alternatives in natural language.

Example: When quantifying a query, a user should be able to employ equivalent forms, such as "over 50", "more than 50", "51 or more".

Reference: Michard, 1982.

Query language is a special form of command language that can be used to request information from a computer.

-1 • Query Language -1

Consider query language dialogue for tasks emphasizing unpredictable information retrieval (as in many analysis and planning tasks), with moderately trained users.

Comment: Guidelines for command language design would apply equally to query languages.

Reference: Ehrenreich, 1981.

See also: Section 3.1.5.

-2 • Natural Organization of Data -2

Design a query language so that it reflects a data structure or organization perceived by users to be natural.

Example: If a user supposes that all data about a particular person are stored in one place, then the query language should probably permit such data to be retrieved by a single query, even though actual computer storage might carry the various data in different files.

Comment: The users' natural perception of data organization can be discovered by survey or experimentation. When users' perceptions do not match the actual structure of computer-stored data, then special care will be needed to preserve the users' viewpoint in query language design.

Reference: Durding, Becker and Gould, 1977.

-20 o Aborting Erroneous Commands -20

If a user makes a command entry error, after the error message has been displayed the user should be able to enter a new command; a user should not be forced to correct and complete an erroneous command.

Comment: In considering a command entry error message, a user may decide that the wrong command was chosen in the first place, and wish to substitute another command instead.

-21 • Reviewing Destructive Commands -21

When command entries may have disruptive consequences, require users to review and confirm a displayed interpretation of the command before it is executed.

See also: 3.5-8, 6.5-18.

-17 • Standard Techniques for Command Editing -17

Users should be able to edit erroneous commands with the same techniques that are employed to edit data entries.

-18 • Interpreting Misspelled Commands -18

Where the set of potential command entries is well defined, program the computer to recognize and execute common misspellings of commands, rather than requiring re-entry.

Comment: This practice should permit a sizable reduction in wasted keying without serious risk of misinterpretation. The necessary software logic is akin to that for recognizing command abbreviations.

Comment: For novice users, it may be helpful for the computer to display an inferred command for user confirmation before execution.

Reference: BB 2.2.4; MS 5.15.7.10; Gade, Fields, Maisano, Marshall and Alderman, 1981.

-19 • Correcting Command Entry Errors -19

When a command entry is not recognized, the computer should give the user a chance to revise the command, rather than rejecting the command outright.

Comment: Misstated commands should not simply be rejected. Instead, software logic should guide users toward proper command formulation. Preserve the faulty command for reference and modification, and do not require a user to re-key the entire command just to change one part.

See also: 3.5-2, 4.3-14.

-14 o Standard Command Delimiter -14

If punctuation other than spaces is needed, perhaps as a delimiter to distinguish optional parameters or to separate entries in a stacked command, adopt a single standard symbol for that purpose.

Example: A slash (/) might be a good choice.

Comment: Whatever symbol is adopted as a delimiter for command entries should preferably be the same as any delimiter that might be used when making data entries.

See also: 1.4-4, 3.2-17.

-15 • Ignoring Blanks in Command Entry -15

The computer should treat single and multiple blanks between words as equivalent when processing command entries.

See also: 1.0-29.

-16 • Abbreviation of Commands -16

Permit users to abbreviate commands.

Example: If a "P" uniquely identifies a print command (i.e., no other commands start with "P") then the user should be able to enter PRINT, or PR, or P, or any other truncation to initiate printing.

Comment: As a corollary, misspelled command entries should also be tolerated, within the limits of computer recognition. The computer can interrogate the user as necessary to resolve ambiguous entries.

Comment: If a command language is still changing, as during system development, do not permit variable abbreviation. For the user, an abbreviation that works one day may not work the next. For the software designer, the addition of any new command might require revision of recognition logic for other commands.

Reference: BB 2.4.3; Demers, 1981.

-11 • Command Stacking -11

Permit users to key a series of commands at one time ("command stacking").

Comment: This practice will allow experienced users to by-pass prompting sequences. Command stacking will reduce the extended memory load on users. Command stacking may also be much faster than separate entry of commands, in systems where input/output processing is overloaded by multiple users.

Reference: BB 2.9.

See also: 3.2-13, 4.4-9.

-12 o User Definition of Macro Commands -12

Permit users to assign a single name to a defined series of control entries, and then use that named "macro" for subsequent command entry.

Comment: In this way users can make frequently required but complicated tasks easier to accomplish, when the interface designer has failed to anticipate a particular need.

Reference: Demers, 1981; Foley and Wallace, 1974.

See also: 3.2-18.

-13 • Minimal Command Punctuation -13

Permit users to enter commands without any punctuation other than the spaces between words.

Reference: MS 5.15.4.5.4.

See also: 3.2-16.

-11 • Indicating Control Defaults -11

When control is accomplished by keyed command or option code entries, if a default command is defined for a null control entry then indicate that default to the user.

Example: "Press ENTER to see more options."

Exception: If a consistent default is adopted throughout interface design, that default need not be explicitly indicated for each individual transaction.

Comment: Here the phrase "null control entry" refers to pressing an ENTER key without first keying a command or option code (and without any accompanying data). It does not refer to defaults for optional parameters that might accompany a valid control entry, whose values might be displayed only at user request.

Comment: It is not necessary that any defaults be defined for null control entries. In such cases, the computer might simply respond, "ENTER alone is not recognized here." The point here is that when defaults are defined, and when they vary from one transaction to another, then users should be informed of the current default logic.

See also: 4.4-7.

-12 • Consistent CONTINUE Option -12

At any step in a defined transaction sequence, if there is only a single appropriate next step then provide a consistent control option to continue to the next transaction.

Example: CONTINUE or NEXT or STEP might be suitable names for this option.

Exception: If data entry is involved, then require a user to select an explicit ENTER option, rather than simply to CONTINUE.

Reference: PR 4.11.

See also: 4.4-5.

-13 • Stacked Commands -13

Permit users to key a sequence of commands or control option codes as a single "stacked" command entry.

Comment: In particular, permit users to enter stacked commands from any menu so that an experienced user can make any specific control entry without having to view subsequent menus.

Comment: Command stacking may be helpful when a user is being prompted to enter a series of parameter values, and knows what several succeeding prompts will request and what values to enter.

Comment: Command stacking will permit a transition from simple step-by-step control entry by novice users, as in menu selection and question-and-answer dialogues, to the entry of extended command-language statements by experienced users; command stacking is especially helpful in time-shared systems where computer response to any user entry may be slow.

Reference: EG 6.2, 6.2.1; PR 2.6, 4.7.3; Palme, 1979.

See also: 3.1.5-11.

-14 o Consistent Order in Command Stacking -14

In command stacking, user entries should be in the same order as they would normally be made in a succession of separate control entry actions.

Reference: EG 6.2.1.

-15 o Abbreviation in Command Stacking -15

In command stacking, accept command names or their abbreviations or option codes just as if those control entries had been made separately.

Comment: In some applications, it might prove helpful if the computer were to display its interpretation of a stacked command for user review and confirmation.

Reference: EG 6.2.1.

-16 o Minimal Punctuation of Stacked Commands -16

Permit users to enter stacked commands without any punctuation other than spaces between words.

Comment: Sometimes stacked commands may require specific symbols as delimiters for their interpretation. Careful design of command languages and/or option codes can minimize the need for delimiters to interpret correct entries. Delimiters may still be needed, however, to protect against possible user errors, i.e., stacked commands that have been improperly composed.

See also: 3.1.5-13.

-17 o Standard Delimiter in Command Stacking -17

If punctuation other than spaces is needed to separate entries in a stacked command, adopt a single standard symbol for that purpose.

Example: A slash (/) may be a good choice.

Comment: Whatever symbol is adopted as a delimiter for command entries should preferably be the same as any delimiter that might be used when making data entries.

Reference: EG 6.2.1.

See also: 1.4-4, 3.1.5-14.

-18 • User Definition of Macro Commands -18

Provide flexibility in transaction selection by permitting users to assign a single name to a defined series of control entries, and then use that named "macro" for subsequent command entry.

Comment: In this way users can make frequently required but complicated tasks easier to accomplish, when the interface designer has failed to anticipate a particular need.

Reference: Demers, 1981; Foley and Wallace, 1974.

See also: 3.1.5-8, 3.1.5-12.

-19 • User-Specified Transaction Timing -19

When appropriate to task requirements, permit users to specify transaction timing, i.e., when a requested transaction should start or should be completed, or the periodic scheduling of repeated transactions.

Example: A user might wish to specify that a requested data analysis routine be deferred until some later hour, to ensure that interim updates to the data will be taken into account.

Example: A user might prepare a number of messages for transmittal, but specify that actual transmission be deferred until a later time.

Example: A user might wish to specify that a request for a large printout be deferred to take advantage of reduced overnight rates, but specify a printout deadline to ensure delivery by 0800 the next morning.

Example: A user might wish to specify that summarized briefing material be prepared automatically at 0600 every morning until further notice.

Comment: In many applications, of course, users will wish specified transactions performed as quickly as possible. In some applications, however, users may have good reasons to delay initiation (or completion) of transactions.

Comment: In some instances, it may be possible to provide appropriate software logic to aid user decisions on transaction timing. For example, if a user requested a bulky printout, the computer might propose overnight printing as an economical alternative, subject to user confirmation.

Comment: Permitting users to specify periodic scheduling for routine transactions will tend to reduce the memory load on users, and may help ensure more reliable system performance. If such routine transactions require further user inputs, as when preparing periodic activity reports, computer logic can be devised to prompt users on a timely basis to make the necessary entries.

For flexibility of sequence control, a user should be able to interrupt and redirect ongoing transactions.

-1 • User Interruption of Transactions -1

Provide flexibility in sequence control by permitting users to interrupt or abort a current transaction, in ways appropriate to task requirements.

Comment: Provision of flexible interrupt capabilities will generally require some sort of suspense file or other buffering in software design. A buffering capability, however, is also helpful for other reasons, permitting users to correct mistaken entries, and permitting the computer to require user confirmation of potentially destructive entries.

Reference: PR 3.3.16, 3.3.17.

-2 • Distinctive Interrupt Options -2

If different kinds of user interrupt are provided, design each interrupt function as a separate control option with a distinct name.

Comment: As a negative example, it would not be good design practice to provide a single INTERRUPT key which has different effects depending upon whether it is pushed once or twice. Users may be confused by such an expedient, and uncertain about what action has been taken and its consequences.

-3 o CANCEL Option -3

If appropriate to sequence control, provide a CANCEL option which will have the effect of erasing any changes just made by the user and restoring the current display to its previous version.

Example: In a sequence of related data entries, on several display frames, CANCEL might erase ("clear") data in the current frame as a convenient way to begin keying corrected data, rather than having to erase each data item individually.

Comment: The easiest way to implement CANCEL may be simply to regenerate the current display.

Reference: Foley and Wallace, 1974.

See also: 1.4-2.

-4 o BACKUP Option -4

If appropriate to sequence control, provide a nondestructive BACKUP option which will have the effect of returning to the display for the last previous transaction.

Example: In a sequence of related data entries, on several display frames, BACKUP might return to the previous frame, where data items could then be erased by CANCEL or could be edited individually.

Comment: Such a BACKUP capability will generally prove feasible only in the software design of well-defined transaction sequences, but will prove helpful when it can be provided.

Comment: In some applications, BACKUP might be designed to include cancellation of any interim entries made in a pending transaction. More often, however, it will be better to preserve pending entries without processing. Interface design should be consistent in this regard.

Reference: MS 5.15.7.7; Foley and Wallace, 1974.

See also: 1.4-2.

-5 o RESTART Option -5

If appropriate to sequence control, provide a RESTART option which will have the effect of returning to the first display in a defined transaction sequence, permitting the user to review a sequence of entries and make necessary changes.

Example: In a sequence of related data entries, on several display frames, RESTART might return to the first frame, from which data could be reviewed and edited as needed throughout the sequence of frames.

Comment: RESTART is an extension of the BACKUP capability, and is useful only in well-defined transaction sequences such as step-by-step data entry in a question-and-answer dialogue.

Comment: In some applications, RESTART might be designed to include cancellation of any interim entries made in a pending transaction. More often, however, it will be better to preserve pending entries without processing. Interface design should be consistent in this regard.

See also: 1.4-2.

-6 o ABORT Option -6

If appropriate to sequence control, provide an ABORT option which will have the effect of canceling any entries that may have been made in a defined transaction sequence and returning to the beginning of the sequence; when data entries or changes will be nullified by an ABORT action, require users to CONFIRM the ABORT.

Example: In a sequence of related data entries, on several display frames, ABORT might erase all data entries in the sequence and return to the first frame.

Comment: An ABORT action combines the functions of RESTART and CANCEL, and is relevant only to well-defined transaction sequences.

Reference: BB 4.7; MS 5.15.7.5.d.

See also: 3.0-18, 4.2-11, 6.5-3.

-7 o END Option -7

If appropriate to sequence control, provide an END option which will have the effect of concluding a repetitive transaction sequence.

Example: In a repetitive sequence of data entries, where completing one transaction cycles automatically to begin the next, END might break the cycle and permit the user to select other transactions.

Comment: END can be implemented by whatever means are appropriate to the dialogue design, i.e., by menu selection, command entry, or function key.

Reference: EG 4.2.10.

-8 o PAUSE/CONTINUE Options -8

If appropriate to sequence control, provide PAUSE (and CONTINUE) options which will have the effect of interrupting (and later resuming) a transaction sequence without any change to data entries or control logic for the interrrupted transaction.

Example: A user might wish to interrupt a current task to read an incoming message.

Example: In the interests of data protection, a user might wish to blank a current display to prevent its being read by some casual visitor.

Comment: A "security pause", as postulated in the second example, may have to be implemented quickly and easily, which suggests that this option should be offered via function key.

See also: 6.2-6.

-9 o Indicating PAUSE Status -9

If a PAUSE option is provided, the computer should display some indication of the PAUSE status whenever that option is selected by a user, and prompt the CONTINUE action that will permit resumption of the interrupted transaction.

-10 o SUSPEND Option -10

If appropriate to sequence control, provide a SUSPEND option which will have the effect of preserving current transaction status when a user leaves the system, and permitting resumption of work at that point when the user later logs back onto the system.

Comment: In the interests of data protection, a SUSPEND option might require special user identification procedures at subsequent log-on, to prevent unauthorized access to suspended transactions.

See also: 6.1-5.

-11 o Indicating SUSPEND Status -11

If a SUSPEND option is provided, the computer should display some indication of the SUSPEND status whenever that option is selected by a user, and at subsequent log-on prompt the user in those procedures that will permit resumption of the suspended transaction.

Designers should ensure that current context which affects the outcome of control actions is evident to users.

-1 • Defining Context for Users -1

Design the sequence control software to maintain context for the user throughout the series of transactions comprising a task; where appropriate, display the results of previous entries affecting present actions, and indicate currently available options.

See also: 1.8-8, 3.0-9, 4.4-10.

-2 o Context Established by Prior Entries -2

Design the sequence control software to interpret current control actions in the context of previous entries; do not require users to re-enter data.

Example: If data have just been stored in a named file, permit users to request a printout of that file without having to re-enter its name.

Exception: If transactions involving contextual interpretation would have destructive effects (e.g., data deletion), then display the interpreted command first for user confirmation.

Comment: The software logic supporting contextual interpretation of control entries need not be perfect in order to be helpful. The computer may occasionally have to present an interpreted command for user review and approval. That is still easier for the user than having to specify every command completely in the first place.

Reference: MS 5.15.2.1.6; PR 2.3.

-3 o Record of Prior Entries -3

Permit users to request a summary of the results of prior entries to help determine present status.

Example: In an aircraft assignment task, there might be a status display showing current commitments of aircraft to missions.

Example: In a text processing application, there might be a status display listing documents already edited and printed in the current work session.

Comment: Summarizing prior entries will be particularly helpful in tasks where transaction sequences are variable, where a user must know what was done in order to decide what to do next. Summarizing prior entries may not be needed for routine transactions if each step identifies its predecessors explicitly, although even in those circumstances a user may be distracted and at least momentarily become confused.

Reference: EG 4.2.7.

See also: 3.1.1-3, 4.4-18.

-4 • Display of Operational Mode -4

When context for sequence control is established in terms of a defined operational mode, remind users of the current mode and other pertinent information.

Example: If text is displayed in an editing mode, then a caption might indicate EDIT as well as the name of the displayed text; if an INSERT mode is selected for text editing, then some further displayed signal should be provided.

Reference: BB 4.3.4; EG 4.2.1; MS 5.15.5.5, 5.15.7.5.a.

See also: 4.1-5, 4.2-8, 4.4-10.

-5 • Display of Control Parameters -5

Permit users to review any control parameter(s) that are currently operative.

Example: A text processing system might display a variety of parameters to control document printing, including margin widths, line spacing, number of copies, etc., which would represent current default values that a user could review and change when desired.

Example: A system might display a "user profile" that specifies for a particular user which editor will be used, which message system, which general menu, what level of prompting, etc.

Comment: A capability for parameter review is helpful even when a user selects all parameters personally. Users will sometimes be forgetful, or may become confused, particularly if their activities are interrupted for any reason.

Reference: MS 5.15.4.1.5.

See also: 4.2-8, 4.4-10.

-6 • Highlighting Selected Data -6

When a user is performing an operation on some selected display item, highlight that item.

Comment: This practice will help avoid error, if a user has misunderstood or perhaps forgotten which item was selected.

Reference: EG 2.1.1.

See also: 4.2-10.

-7 • Consistent Display of Context Information -7

Ensure that information displayed to provide context for sequence control is distinctive in location and format, and consistently displayed from one transaction to the next.

Reference: MS 5.15.4.4.

See also: 4.0-6.

> Users will make errors, and interface design must facilitate the detection and correction of those errors.

-1 • Appropriate Response to All Entries -1

The computer software should deal appropriately with all possible control entries, correct and incorrect.

Example: If the user selects a function key that is invalid for a particular transaction, no action should result except display of an advisory message indicating what functions are appropriate at that point.

Comment: For certain routine and easily recognized errors, such as trying to tab beyond the end of a line, a simple auditory signal ("beep") may be sufficient computer response.

Reference: PR 4.12.4.5.

See also: 3.1.4-10, 6.0-5.

-2 • Command Editing -2

Permit users to edit an extended command during its composition, by backspacing and rekeying, before taking an explicit action to ENTER the command.

Comment: Users can often recognize errors in keyed entries prior to final entry.

Reference: EG 5.4; MS 5.15.7.2; Neal and Emmons, 1982.

See also: 1.4-2, 3.1.5-19, 6.0-6, 6.3-10.

-3 • Prompting Command Correction -3

If an element of a command entry is not recognized, or logically inappropriate, prompt users to correct that element rather than requiring re-entry of the entire command.

Example: A faulty command can be retained in the command entry area of the display, with the cursor automatically positioned at the incorrect item, plus an advisory message describing the problem.

Reference: BB 5.2.1; EG 4.2.2, 4.2.3; MS 5.15.7.1.

See also: 4.3-14.

-4 • Errors in Stacked Commands -4

If an error is detected in a stacked series of command entries, either consistently execute to the point of error, or else consistently require users to correct errors before executing any command.

Comment: It may help the user if the commands are executed to the point of error, or it may not. In most applications, partial execution will probably prove desirable. The point here is that a considered interface design decision should be made and then followed consistently.

Reference: EG 5.6; PR 4.7.3.

-5 o Partial Execution of Stacked Commands -5

If only a portion of a stacked command can be executed, notify the user and provide appropriate guidance to permit correction, completion, or cancelation of the stacked command.

Comment: Note that stacked commands can fail because of error in their composition, or for other reasons such as unavailability of required data.

Reference: MS 5.15.7.11; Dwyer, 1981.

See also: 4.4-7.

User guidance refers to error messages, alarms, prompts, and labels, as well as to more formal instructional material.

-1 • Standard Procedures -1

Design standard procedures for accomplishing similar, logically related transactions, in order to facilitate user learning and efficient system operation.

Comment: A designer may argue that for one particular transaction a standard procedure does not seem efficient. Perhaps the standard procedure requires one or two more keystrokes than some special procedure that might be devised. But every special feature of interface design will put a small added burden on the user's memory, and where special procedures are not remembered they may not be used properly. Standard procedures will increase overall operational efficiency.

Reference: *BB* 1.2.1, 2.1.5; Reisner, 1981.

See also: 3.0-6, 3.0-7, 6.0-2, 6.5-6.

Objectives:

Consistency of operational procedures
Efficient use of full system capabilities
Minimal memory load on user
Minimal learning time
Flexibility in supporting different users

Guidelines:

- User guidance refers to error messages, alarms, prompts, and labels, as well as to more formal instructional material.
- Information on current data processing status should be available to users at all times.
- The computer should provide feedback to its users as transactions are routinely processed.
- When an error or other unexpected event prevents routine transaction processing it is important to inform the user.
- Specific task-oriented guidance should be provided throughout the transaction sequences that comprise a user's job.
- Many aspects of interface design, and user guidance, can be assessed and improved by recording user performance.
- Changes to the software design of user guidance functions may be needed to meet changing operational requirements.

shortcut provided by software designers will add an extra paragraph, or sentence, or footnote to the instructional material. As special features proliferate, instructions to the user must expand accordingly. On-line guidance will require more display space, and printed manuals will grow fatter.

From this reasoning, we may conclude that interface design may be improved if its documentation begins early, when changes are still relatively easy to make. On-line documentation can certainly be prepared (and changed) more quickly than printed user manuals, which suggests that early on-line documentation may help interface designers as well as the eventual system users.

Preparation of user guidance material will serve to indicate the point of diminishing returns in further elaboration of user interface software. If it takes ten minutes for a user to learn a special procedure that might save ten seconds in a seldom-used transaction, then design elaboration has outpaced user needs. Considering questions of user guidance throughout interface design will ensure that a sensible balance is struck between efficiency of operation and ease of learning, between functionality and usability. This practical trade-off has been noted by others concerned with user interface design (e.g., Goodwin, 1982), but requires continuing emphasis.

In considering guidelines for design of user guidance functions, we must recognize that user guidance is a pervasive concern in interface design. Many of the guidelines proposed for data entry, data display, and sequence control functions have the implicit objective of making a system easier to learn and easier to understand during its use. From general recommendations for design consistency, through more detailed guidelines to help distinguish different aspects of information handling transactions, there must be an underlying concern for guiding the user's interaction with the automated system. Thus almost any guideline for user guidance could be cross-referenced to a wide range of other design recommendations. Of the many possible cross references, a number of specific interest are cited in the guidelines proposed here.

guidance will adapt the interface to different user capabilities, supporting the novice user without hindering the expert.

The concern here is with on-line user instruction, as opposed to off-line system documentation. The need for on-line instruction is emphasized by Brown, Brown, Burkleo, Mangelsdorf, Olsen and Perkins in their user interface guidelines developed as an in-house design standard at Lockheed:

> Much of the information commonly provided in paper documentation, such as user manuals, should also be available on line. A manual may not be available when it is needed. Some users may never receive the relevant documentation. They may not know what documents are available, which ones are relevant, or how to procure them. Even users who possess the appropriate documentation will not necessarily have it with them when it is needed.
>
> (1983, page 6-1)

And, one might add, even if users have the appropriate documentation on hand, they may not be able to find answers to their questions, especially when the documentation is bulky. Effective on-line instruction and other forms of user guidance can reduce the need for off-line training courses based on system documentation.

Certainly there is a strong trend in current system design toward on-line documentation for user instruction. This trend reflects the decreasing cost of computer memory, and also the increasing use of computers by people with little understanding of computer function. The novelty of early pioneering efforts to program computers to teach their own use (Morrill, 1967; Morrill, Goodwin and Smith, 1968; Goodwin, 1974) has now become almost commonplace, as we have learned more about the techniques of on-line aiding.

One of the principal arguments for on-line documentation is economic. Limanowski (1983) estimates a potential cost saving of 70 to 80 percent if on-line documentation can be provided as an alternative to more traditional paper documentation. If true, we can expect to see increasing recourse to on-line documentation for that reason alone.

People who are responsible for writing instructional material may be the first to note inconsistencies in user interface design. In the documentation of user guidance, whether intended for on-line display or printed handbooks, every special feature and smart

Consistency is important in all aspects of user interface design. Anyone who has tried to learn a foreign language knows the difficulty of learning irregular verbs, whose conjugation does not follow any consistent rule. In the same way it is difficult to learn (and remember) irregular features of user interface design.

Data entry can be guided by consistent formatting of entry fields on the display, including consistent wording of labels, consistent placement of labels with respect to entry fields, and consistent demarcation of the fields themselves. Some kind of highlighting is frequently recommended for field delineation, displaying field location and boundaries clearly to a user. Even more guidance could be provided by consistent use of different field marking to indicate different types of data entry.

For data display, formats should be consistent from one frame to another, always including a title at the top, labels to indicate page numbers in related displays, standard labeling of control options, standard positioning of guidance messages, etc. Messages indicating user errors should be carefully worded to be both concise and informative. They should also be consistently worded from one message to the next, and consistently located in the display format. Even such a subtle feature as cursor positioning should receive consistent treatment in the software logic of user interface design.

For sequence control, consistent user interface design is even more important. One design feature that can help guide the user is consistent provision of a general OPTIONS display, a "home base" to which the user can return from any point in a transaction sequence in order to select a different transaction. As a basic principle of user guidance, the interface designer should not rely on a user to remember what prior actions have been taken, or even what action is currently being taken. Users may be distracted by competing job demands. For extended transaction sequences, the designer should make available a cumulative record of control actions that can be displayed for user review at user request. For control actions whose consequences are contingent on context, an indication of context should always be displayed, even when that context was defined initially by the user.

Although consistent interface design will provide much inherent guidance to the user, it is often desirable to include in computer-generated displays some explicit instructions or prompts for the user. Such instructions should be consistently located in display formatting, perhaps always at the bottom where they can be ignored by experienced users. For novice users it may be desirable to provide supplementary guidance or job instruction, optionally available in response to user requests for help. Such optional

> Feedback to user action covers keeping the user informed of where he is, what he has done, and whether it was successful. . . . In an interactive computing situation, immediate feedback by the system is important in establishing the user's confidence and satisfaction with the system. One of the more frustrating aspects of any interactive system is sitting at the terminal after entering something and waiting for a response. Questions arise such as, 'Is the system still going?', 'Is the terminal still connected to the system?', 'Did the computer lose my input?', 'Is the system in a loop?'. A message that indicates that the system is still working on the problem or a signal that appears while the system is processing the user's input provides the user with the necessary assurance that everything is all right.
>
> (1975, page 13)

Predictability of machine response is related to system response time. Timely response can be critical in maintaining user orientation to the task. If a response is received only after a long delay, the user's attention may have wandered. Indeed, the user may forget just which action the machine is responding to. Frequent user actions, generally those involving simple inputs such as function key or menu selections, should be acknowledged immediately. In transactions where output must be deferred pending the results of computer search and/or calculation, the expected delay should be indicated to the user in a quick interim message.

Some experts argue that consistency of system response time may be more important in preserving user orientation than the absolute value of the delay, even suggesting that designers should delay fast responses deliberately in order to make them more consistent with occasional slow responses. Perhaps such a reduction in response time variability may be desirable. If system response time is always slow, a user can adapt to the situation and find something else useful to do while waiting. But surely a better solution is to make all responses uniformly fast, or, where that is not possible, to provide a quick interim message to warn the user than a response will be delayed, as suggested above. In that way, a slow response is made predictable even though it is not consistent with other responses.

Ensuring fast response is probably a greater problem in the design of general-purpose, multi-user, time-shared systems than for dedicated information system applications. Where a system is designed to accomplish defined tasks in a specified manner, data processing loads can usually be anticipated sufficiently well in user interface design to provide adequately fast response for all transactions.

SECTION 4

USER GUIDANCE

User guidance refers to error messages, alarms, prompts, and labels, as well as to more formal instructional material provided to help guide a user's interaction with a computer. The fundamental objectives of user guidance are to promote efficient system use (i.e., quick and accurate use of full capabilities), with minimal memory load on the user and hence minimal time required to learn system use, and with flexibility for supporting users of different skill levels.

User guidance should be regarded as a pervasive and integral part of interface design that contributes significantly to effective system operation. User guidance cannot be merely a decoration added at the end, like frosting on a cake. A study by Magers (1983) has demonstrated convincingly that good user guidance can result in faster task performance, fewer errors, greater user satisfaction, and will permit accomplishment of information handling tasks otherwise impossible for novice users.

A narrow view of user guidance deals with preventing and correcting user errors. But minimizing user errors may require improvements in broad aspects of interface design -- in techniques for data display, in procedures for data entry and sequence control -- as well as provision of user guidance. Moreover, any general consideration of user guidance functions must include provision of status information, job aids, and routine feedback, as well as feedback for error correction.

Many user interface design features contribute directly or indirectly to guide a user's interaction with an on-line computer system. The primary principle governing this aspect of interface design is to maintain consistency. With consistent interface design, users can learn to apply computer tools more quickly, more accurately, and with more confidence.

Design consistency implies predictability of system response to user inputs. A fundamental rule is that some response be received. For every action (input) by the user there should be a noticeable reaction (output) from the computer. It is this feedback linking action to reaction that defines each discrete transaction and maintains user orientation in interaction with the system.

The importance of feedback information for guiding the user has been emphasized by Engel and Granda in their recommendations for user interface design:

Changes to the software design of control functions may be needed to meet changing operational requirements.

-1 • Flexible Design for Sequence Control -1

When sequence control requirements may change, which is often the case, provide some means for the user (or a system administrator) to make necessary changes to control functions.

Comment: Sequence control features that may need to be changed include those represented in these guidelines, namely, the types of dialogue that are provided, procedures for transaction selection and user interrupt, methods for context definition and error management, and alarm logic.

See also: 3.0-1, 3.1.3-26, 3.1.5-8, 3.1.5-12, 3.2-18, 3.2-19, 3.6-1.

-4 o Alarm Reset -4

Provide users with a simple means of turning off an auditory alarm, without erasing any displayed message that accompanied the auditory signal.

Example: A function key labeled ALARM RESET would suffice for this purpose.

Comment: Turning off an auditory alarm will ensure that any succeeding alarm condition will be able to generate a new auditory signal to attract the user's attention.

-5 o Acknowledgment of Critical Alarms -5

Users may be required to acknowledge special or critical alarms in special ways; but such special acknowledgment actions should be designed so that they will not inhibit or slow remedial user response to a critical initiating condition.

Users may need to control the logic and operation of alarms and alerting signals.

-1 • Alarm Definition by Users -1

In many applications, particularly those involving monitoring and process control, it will be desirable to permit users to define the conditions (in terms of variables and values) that will result in automatic generation of alarm messages.

Example: The nurse in charge of an intensive care monitoring station might need to specify for each patient that a warning be signaled when blood pressure (a "variable") exceeds or falls below defined levels ("values").

Exception: There are some situations where alarm conditions must be pre-defined by functional, procedural, or perhaps even legal requirements, such as violation of aircraft separation in air traffic control.

See also: 4.1-10.

-2 • Distinctive and Consistent Alarms -2

Ensure that alarm signals and messages are distinctive for each class of events.

Comment: Users should participate in the classification of alarming events, and might help in specifying the desired nature of different alarm signals.

See also: 4.3-17.

-3 • Alarm Acknowledgment -3

Provide users with a simple means of acknowledging and turning off non-critical alarm signals.

Example: A function key labeled ALARM OFF would suffice for this purpose.

-11 o Preventing Data Loss at LOG-OFF -11

When a user requests LOG-OFF, sequence control software should check pending transactions and, if any pending transaction will not be completed, or if data will be lost, should display an advisory message requesting user confirmation.

Example: CURRENT DATA FILES HAVE NOT BEEN FILED;
SAVE IF NEEDED, BEFORE CONFIRMING LOG-OFF.

Comment: The user may sometimes suppose that a job is done before taking necessary implementing actions.

Reference: BB 4.8; MS 5.15.7.5.e.

See also: 4.3-15, 6.5-16.

-12 o Immediate Data Correction -12

When a data entry transaction has been completed and errors detected, sequence control logic should permit users to make corrections directly and immediately.

Comment: It is helpful to correct data entry errors at the source, i.e., while a user still has the entry in mind and/or source documents at hand. When a user cannot correct an entry, as when transcribing from a source document that itself contains an error, it may help to allow the user to defer entry of the wrong item. Alternatively, the user might wish to abort the transaction.

Comment: For transactions involving extended entry of multiple items, computer checking might be invoked as each page (or section) of data is entered.

Reference: EG 5.7; PR 2.5.

See also: 1.7-4, 1.7-6, 3.3-5, 6.3-11.

-13 o Flexible BACKUP for Error Correction -13

Permit users to return easily to previous steps in a transaction sequence in order to correct an error or make any other desired change.

Reference: MS 5.15.7.7.

See also: 3.3-4, 6.3-13.

-9 o Distinctive CONFIRM Action -9

Provide an explicitly labeled CONFIRM function key, different from the ENTER key, for user confirmation of a control entry or data entry.

Comment: Confirmation should not be accomplished by pushing some other key twice.

Comment: Some interface designers recommend that in special cases confirmation should be made more difficult still, e.g., by keying the separate letters C-O-N-F-I-R-M. Even such extreme measures, however, cannot guarantee that users will not make errors.

See also: 3.1.4-4, 3.1.4-7, 6.5-19.

-10 • UNDO to Reverse Control Actions -10

Any user action should be immediately reversible by an UNDO command.

Example: *If a user is overhasty in confirming a destructive action, and realizes the mistake right away (before taking another action), then an UNDO action might be taken to reverse the damage.*

Comment: UNDO itself should be reversible, so that a second UNDO action will do again whatever was just undone.

Comment: Such an UNDO capability is currently available in some interface designs, and should be provided more generally. Even with an UNDO capability, however, a user may make an irretrievable mistake, if succeeding actions intervene before a prior destructive action is noticed.

Reference: Lee and Lochovsky, 1983; Nickerson and Pew, 1971; Shneiderman, 1982.

See also: 6.5-20.

-6 • Explicit Entry of Corrections -6

When a user completes correction of an error, whether of a command entry or data entry, require the user to take an explicit action to re-enter the corrected material; use the same ENTER action for re-entry as was used for the original entry.

Reference: MS 5.15.7.9; PR 4.12.4.6.

See also: 1.0-8, 3.0-5, 6.0-3, 6.3-14.

-7 • User Confirmation of Destructive Entries -7

When a control entry will cause any extensive change in stored data, procedures and/or system operation, and particularly if that change cannot be easily reversed, notify users and require confirmation of the action before implementing it.

Reference: BB 5.6; EG 4.1.2, 4.2.8; MS 5.15.7.4, 5.15.7.5.b; Foley and Wallace, 1974.

See also: 4.3-16, 6.5-14, 6.5-18.

-8 o User Warned of Potential Data Loss -8

Word the prompt for a CONFIRM action to warn users explicitly of any possible data loss.

Example: (Good) CONFIRM DELETION OF ENTIRE AIRFIELD FILE

(Bad) CONFIRM DELETE ACTION

See also: 3.1.5-21, 4.3-15, 6.5-17.

-2 • Explicit User Actions -2

Require users to take explicit actions to specify computer data processing; the computer should not take extra (and possibly unrecognized) actions beyond those specified by a user.

Exception: Automatic cross file updating following data change might be considered an exception to this rule.

Comment: Explicit actions, even though they may require an extra keystroke or two, will help a user to learn procedures and to understand better what is happening in any transaction. In effect, requiring the user to take action to accomplish something can be regarded as a form of guidance.

Comment: An interface designer, with expert knowledge of the system and its internal workings, is sometimes tempted to provide the user with "smart shortcuts", where the computer will execute automatically some action that the user would surely need to take. Incorporating such smart shortcuts in interface design, though done with the intention of helping the user, will risk confusing any but the most expert users.

Reference: MS 5.15.2.1.4.

See also: 1.0-8, 1.7-4, 3.0-5, 3.1.4-7, 3.2-1, 6.0-3.

-3 • Separate LOG-ON Procedure -3

In applications where users must log on to the system, design LOG-ON as a separate procedure that is completed before a user is required to select among any operational options.

Comment: Ideally the entire LOG-ON sequence including option selections will be simple, but in some operational situations that may not be feasible.

Comment: Separate LOG-ON will focus user attention on the required input(s), without the distraction of having to anticipate other decisions, and will help reduce initial confusion, particularly for novice users.

Reference: BB 4.6.1.

-4 • Display of Guidance Information -4

In general, follow recommendations for the design of data displays when designing user guidance displays.

Comment: Some of the specific guidelines for data display are restated for convenient reference in this section, as particularly appropriate for display of user guidance. Many other applicable data display guidelines are cited by cross reference.

See also: Section 2.

-5 • Only Necessary Information Displayed -5

Tailor the display for any transaction to the current information requirements of the user, so that only relevant data are displayed.

Comment: When this can be done successfully, so that only relevant data are displayed, the display itself provides implicit guidance, showing what data should be considered. Conversely, display of irrelevant data will tend to confuse the user.

Reference: BB 1.7; MS 5.15.3.1.2.

See also: 2.0-1, 2.0-2, 2.2-1, 2.2-2, 2.6-1.

-6 • Consistent Display Format -6

Create display formats with a consistent structure evident to the user, so that different types of data are always presented in the same places and in the same ways.

Comment: Consistent display formats will help new users learn to interact efficiently with the system.

Reference: EG 2.3; MS 5.15.3.1.1.

See also: 1.4-22, 1.4-23, 1.5-4, 2.3-1, 2.5-4, 3.0-8, 3.1.3-8, 3.1.3-30, 3.1.5-2, 3.4-7, 4.4-8.

-7 o Consistent Format for User Guidance -7

Format each different type of user guidance consistently across displays.

Example: Display titles might be centered at the top of the display, with display identification codes at the upper left corner. The bottom line of the display should be reserved for command entries, where needed, in which case the line just above it could be used for prompts and advisory messages.

Comment: Types of user guidance include display titles, labeling of data entry fields, prompts for data/command entry, error messages, alarms, status and other advisory messages, as well as on-line instructional material.

Comment: Consistent allocation of particular areas of a display for user guidance may be sufficient. Certain types of guidance, however, such as alarms and error messages, may require auxiliary coding to help attract user attention.

Reference: BB 1.1.1, 1.1.2, 2.1.2; EG 2.3; PR 4.5.3.

See also: 1.4-15, 2.3-10, 4.0-6, 4.0-8, 4.3-16.

-8 • Distinctive Format for User Guidance -8

Design display formats so that user guidance material is readily distinguishable from displayed data.

Comment: Consistent location of user guidance on the display will usually suffice, but other formatting conventions may help distinguish particular categories of user guidance, such as labels, prompts, etc., as recommended in other guidelines.

Reference: BB 1.8.5, 2.1.1; EG 2.1, 2.3, 3.2.3.

See also: 1.4-14, 1.5-2, 2.1.2-8, 2.3-2, 3.1.3-18, 3.1.4-15.

-9 o Distinctive Cursor -9

Design cursors (in terms of shape, blink, or other means of highlighting) so that they are readily distinguished from other displayed items.

Comment: When a cursor is automatically positioned under computer control, it can serve to direct the user's attention to a particular point on a display, and so it should be designed to catch the eye. Even when the cursor has been positioned by a user, if the user is momentarily distracted then a distinctive format may help locate the cursor.

Comment: A cursor is the most immediate and continuously available form of user guidance, since it will generally mark the current focus of user attention. With this in mind, the interface designer may decide to use different cursor formats to denote different operational conditions. If that is done, each of those different cursors should be distinctive from other displayed items, and from each other.

See also: 1.1-1, 4.1-5.

-10 • Clear Control Labels -10

Label function keys and other controls clearly to indicate their function.

Reference: BB 4.4.7; MS 5.15.2.3.9.

See also: 1.0-9, 3.1.4-4.

-11 • Clear Data Labels -11

Label all displayed data clearly.

Comment: Labels for individual data fields can be omitted only where display format and labeling of grouped data clearly identify subordinate items, as in row/column labeling of tabular data.

Reference: BB 1.8.7; MS 5.15.3.1.9.

See also: 1.4-5, 1.4-17 thru -19, 1.4-21, 1.5-3, 2.1.2-3.

-12 • Highlighting Critical User Guidance -12

Adopt whatever techniques are used to highlight critical items in data display also to add emphasis to the display of critical user guidance information.

Comment: Alarms and warning messages may require output of auxiliary auditory signals as well as display highlighting, to help assure that they attract the user's attention.

Reference: EG 2.1; MS 5.15.3.3.1.

See also: 2.4-1, 3.6-2, 4.3-17.

-13 • Consistent Coding Conventions -13

Symbols and other codes should have consistent meanings from one display to another.

Comment: This practice will aid user learning of new codes, so that they will gain familiarity. Where codes have special meanings, those should be defined in the display.

Reference: BB 3.6.1.

See also: 2.4-6, 2.4-15, 2.4-29, 3.1.3-12, 4.4-17.

-14 o Familiar Coding Conventions -14

Codes and abbreviations for data entry/display should conform to conventional usage and user expectations.

Comment: Conventional usage will aid user learning of codes, and reduce the likelihood of user error in both code generation (entry) and interpretation (display).

Comment: Deviation from familiar meanings, such as using an aircraft symbol to denote artillery and vice versa, would almost certainly confuse users.

Reference: BB 3.7.1.

See also: 2.4-4, 2.4-29.

-15 • Consistent Wording -15

Make the names for function keys, command names, etc., consistent for similar or identical functions in different transaction sequences.

Example: As a negative example, do not call the same function EDIT in one place, MODIFY in another, UPDATE in a third.

Comment: Consistency in interface design is the fundamental basis of effective user guidance.

Reference: BB 3.7.2; EG 4.2.9.

See also: 3.0-10, 3.1.3-11, 3.1.3-13, 3.1.3-27, 3.1.5-6.

-16 o Familiar Wording -16

Adopt terminology familiar to users when wording labels, prompts and user guidance messages.

Example: (Good) Data requires special access code; call Data Base Admin, X 9999.

(Bad) IMS/VS DBMS private data; see DBSA, 0/99-99.

Comment: User testing and iterative design will often be needed to eliminate difficult words, abbreviations and acronyms that are not generally familiar to all users.

Reference: BB 3.7.1, 3.7.3; EG 3.4.5, 4.2.12; PR 2.4; Magers, 1983.

See also: 2.0-4, 2.0-11, 3.1.5-5.

-17 o Task-Oriented Wording -17

Adopt task-oriented wording for labels, prompts and user guidance messages, incorporating whatever special terms and technical jargon may be customarily employed in the users' tasks.

Comment: Jargon terms may be helpful, if they represent the jargon of the user and not of the designer or programmer. The rule here should be to know the users and adapt interface design to their vocabulary instead of forcing them to learn new wording.

Reference: BB 3.7.3; EG 4.2.13; PR 2.4; Magers, 1983.

See also: 1.4-20, 2.0-11, 3.1.5-5, 3.1.5-6, 3.2-9.

-18 o Affirmative Statements -18

Adopt affirmative rather than negative wording for user guidance messages.

Example: (Good) Clear the screen before entering data.

(Bad) Do not enter data before clearing the screen.

Comment: Affirmative statements are easier to understand. Tell the user what to do, rather than what to avoid.

Reference: BB 3.8.3, 5.3.9.

See also: 2.1.1-13.

-19 o Active Voice -19

Adopt active rather than passive voice in user guidance messages.

Example: (Good) Clear the screen by pressing RESET.

(Bad) The screen is cleared by pressing RESET.

Comment: Sentences in active voice are easier to understand.

Reference: BB 3.8.5.

See also: 2.1.1-14.

-20 o Temporal Sequence -20

When user guidance describes a sequence of steps, follow that same sequence in the wording of user guidance messages.

Example: (Good) Enter LOG-ON sequence before running programs.

(Bad) Before running programs, enter LOG-ON sequence.

Reference: BB 3.8.6.

See also: 2.1.1-15.

-21 o Consistent Grammatical Structure -21

Be consistent in grammatical construction when wording user guidance.

Example:

(Good)	(Bad)
Options:	Options:
s = Select data	s = Select data
e = Erase display	d = Erasure function
w = Write file	w = Write file

Comment: Even minor inconsistencies can distract a user, and delay comprehension as the user wonders momentarily whether some apparent difference represents a real difference.

Comment: Consistent grammatical construction may help a user resolve an ambiguous message (e.g., "Numeric entry"), to understand whether it recommends an action ("You should enter a number") or indicates an error condition ("You entered a number when you shouldn't have").

Reference: BB 3.8.4; Smith, 1981b.

See also: 2.1.2-5, 3.1.3-10.

-22 • Flexible User Guidance -22

When techniques adopted for user guidance (display of option lists, command prompting, etc.) may slow an experienced user, provide alternative paths/modes to by-pass standard guidance procedures.

Comment: Multiple paths, such as command entry to by-pass a menu, or use of abbreviated rather than complete commands, can speed the performance of an experienced user. The interface designer, however, should take care that such shortcuts supplement rather than supplant the standard, fully guided procedures provided for novice users.

Reference: BB 4.5.

See also: 3.0-2, 4.4-26, 4.4-27.

-23 o Easy Ways to Get Guidance -23

Permit users to switch easily between any information handling transaction and its associated guidance material.

Example: Guidance might be displayed as a temporary "window" overlay on the working display, which a user could request or suppress at will.

Comment: If user guidance is difficult to obtain, and/or if asking for guidance will disrupt a current transaction (e.g., erase a working display), then users will prefer to guess at proper procedures rather than seeking help.

Reference: Limanowski, 1983.

Information on current data processing status should be available to users at all times, automatically or by request.

-1 • Indicating Status -1

Provide some indication of system status to users at all times.

Comment: In some applications, system status may be continuously displayed. Status display can be explicit (e.g., by message), or can be implicit (e.g., by a displayed clock whose regular time change offers assurance that the computer link is still operating). Alternatively, system status information might be provided only on user request, following a general or specific query.

Comment: Status information is particularly needed, of course, when system operation is unreliable for any reason. Under those conditions, if status information is not provided by design, users will often devise their own repertoire of harmless but time wasting test inputs to check system performance.

Comment: When system status changes, it may be desirable for the computer to generate an advisory message to draw users' attention to that change.

Reference: BB 4.3; MS 5.15.1.4.b.

-2 • Automatic LOG-ON Display -2

In applications where users must log on to the system, display appropriate prompts for LOG-ON procedures automatically at a user's terminal; do not require users to take any special action to obtain a LOG-ON display, other than turning on the terminal.

Comment: An automatic LOG-ON display will signal the operational availability of a terminal, as well as prompting the user to make necessary initial inputs.

Reference: BB 4.6.1; EG 4.2.6.

See also: 4.0-3.

-3 o LOG-ON Delay -3

If a user tries to log on to a system, and LOG-ON is denied, display an advisory message to tell the user what the system status is and when the system will become available.

Example: "System is down for maintenance until 9:30 AM."

Comment: Avoid "as soon as possible" messages. Make an estimate of system availability, and update the estimate later if that becomes necessary.

Reference: BB 4.3.5.

-4 • Keyboard Lock -4

If at any time the keyboard is locked, or the terminal is otherwise disabled, notify the user.

Example: Control lockout could be signaled by disappearance of the cursor from the display, or by a notable change in the shape of the cursor, accompanied by an auditory signal.

Comment: An auditory signal will be especially helpful to touch typists, who may be looking at source documents for data entry rather than at display or keyboard.

See also: 3.0-17.

-5 • Operational Mode -5

When the results of user action are contingent upon different operational modes, then clearly indicate the currently selected mode.

Example: A change in display caption and/or cursor shape might suffice to alert users to changing modes.

Reference: BB 4.3.4; MS 5.15.7.5.b.

See also: 3.4-4, 4.2-8, 4.4-10.

-6 • Other Users -6

In applications where task performance requires data exchange and/or interaction with other users, permit users to obtain status information concerning other people currently using the system.

Comment: If there are many other users, it may be helpful to allow a user to ask whether any particular individual is a current user.

See also: 5.3-2.

-7 • System Load -7

In applications where task performance is affected by operational load (e.g., number of on-line users), permit users to obtain status information indicating current system performance, expressed in terms of computer response time.

Comment: It may be necessary to define a "standard" function for which computer response time is predicted on a normalized basis.

Comment: Such load information is primarily helpful, of course, when system use is optional, i.e., when a user can choose to defer work until low-load periods. But load status information may help in any case by establishing realistic user expectations for system performance.

-8 • External Systems -8

In applications where task performance requires data exchange and/or interaction with other systems, permit users to obtain relevant status information for external systems.

See also: 5.3-2.

-9 • Date and Time Signals -9

In applications where task performance requires or implies the need to assess currency of information, annotate displays with date-time signals.

Comment: Date-time status might be displayed continuously or periodically, as on displays that are automatically updated, or by user request, depending on the application.

-10 • Alarm Settings -10

In applications where alarm signals are established on the basis of logic defined by users, permit users to obtain status information concerning current alarm settings, in terms of dimensions (variables) covered and values (categories) established as critical.

Comment: Alarm status information will be particularly helpful in monitoring situations where responsibility may be shifted from one user to another ("change of watch").

See also: 3.6-1.

The computer should provide feedback to its users as transactions are routinely processed.

-1 • Consistent Feedback -1

Every input by a user should consistently produce some perceptible response output from the computer; when a terminal is in use, its display screen should never be blank.

Exception: A user might choose to blank a display screen temporarily, perhaps to protect data from casual onlookers, but even then some acknowledging message should probably appear, e.g., DISPLAY TEMPORARILY SUPPRESSED.

Comment: Keyed entries should appear immediately on the display. Function key activation or command entries should be acknowledged either by evident performance of the requested action, or else by an advisory message indicating an action in process or accomplished. Inputs that are not recognized by the computer should be acknowledged by an error message.

Comment: Absence of system response is not an effective means of signaling acceptable entry. At best, a dialogue without feedback will be disconcerting to the user, as when we talk to an unresponsive human listener. At worst, the user may suspect system failure, with consequent disruption and/or termination of the interaction sequence.

Reference: BB 4.3, 4.3.3, 5.1; EG 3.3.2, 4.2.5, 6.3.7; MS 5.15.2.1.2, 5.15.4.1.12, 5.15.4.1.13; Magers, 1983.

See also: 1.0-2, 1.0-11, 1.0-12, 3.0-11, 3.0-13, 3.1.3-9, 3.1.4-8, 6.2-6.

-2 • Fast Response -2

Computer response to user entries should be rapid, with consistent timing as appropriate to different types of transactions.

Reference: MS 5.15.1.8; Stewart, 1980.

See also: 1.1-5, 2.5-5, 3.0-15, 3.1-2.

-3 • Feedback for Control Entries -3

The computer should provide some indication of transaction status whenever the complete response to a user entry will be delayed.

Comment: After making an entry to the computer, the user needs feedback to know whether that entry is being processed properly. Delays in computer response longer than a few seconds can be disturbing to the user, especially for a transaction that is usually processed immediately. In such a case some intermediate feedback should be provided, perhaps as an advisory message that processing has been initiated, and ideally with an estimate of how long it will take to complete.

Comment: Note that a routine advisory (e.g., WAIT FOR PROCESSING) displayed before every computer response, whether fast or slow, is not an effective indication of transaction status. Users will come to ignore routine messages that are sometimes true but sometimes just false alarms.

Reference: BB 4.3.1; EG 4.2.5; MS 5.15.2.1.3.

See also: 3.0-11.

-4 o Indicating Completion of Processing -4

When computer processing of a user entry has been delayed, inform the user when processing is completed, and provide appropriate guidance for further user actions.

Comment: For long delays, interim feedback on processing status before completion may be reassuring to a user. Such follow-up messages, however, should not interfere with current user activities. It may be desirable to reserve a special display window for prompts and advisory messages.

Reference: BB 4.3.2; MS 5.15.5.3.

See also: 3.0-12.

-5 • Feedback for Print Requests -5

When user requests for printed output will be handled by a remote printer, give the user an advisory message confirming that a print request has been received.

Reference: EG 4.2.14.

See also: 1.3-30.

-6 • Display Identification -6

Provide a unique identification in a consistent location at the top of each display page.

Exception: As a possible exception, interface designers may sometimes provide unlabeled displays as "free-form" screens for text entry and other tasks involving display composition by users.

Comment: A displayed title may suffice, although a shorter identification code may be helpful for some purposes. The objective is to help the user recognize a display when it appears, to learn interactive sequences stepping from one display to another, and (in some system applications) to request a particular display directly. Display identification will also help both users and interface designers to refer to individual displays in discussion and documentation.

Comment: In applications involving menu selection, it may prove helpful to code each display with the string of option selections (letter codes) used to reach that display. This practice is particularly useful in situations where a user can learn to by-pass the menu selection sequence by entering option string codes as a single command to request a familiar data display.

Reference: BB 1.2.3; MS 5.15.3.1.9; PR 4.5.2.

See also: 2.3-9, 2.5-2 thru -4.

-7 o Identifying Multipage Displays -7

When lists or data tables extend beyond the capacity of a single display frame, inform the user that the display is continued in multiple frames.

Example: Incomplete lists might be annotated at the bottom as CONTINUED ON NEXT PAGE.

Example: For extended data tables, the display title might be annotated PAGE ____ OF ____.

Example: For scrolled material, displays might be annotated with the current and concluding locations, LINE ____ OF ____.

Exception: In special formats such as spreadsheets, the partial nature of a display may be self-evident.

Comment: As a complementary recommendation, it may also be desirable to conclude completed lists with the annotation END OF LIST, unless the list is so short that it obviously does not fill available display space.

Reference: BB 1.9.7; EG 3.4.1; PR 4.5.5.

See also: 2.6-5, 2.6-6.

-8 • Indicating Operational Mode -8

When a user (or computer) action establishes a change in operational mode that will affect subsequent user actions, display some continuing indication of current mode.

Example: Selection of a DELETE mode in text editing should produce some kind of warning signal on the display, perhaps by a distinctive change in cursor shape.

Comment: This practice is particularly helpful when the mode selected is one seldom used.

Comment: Display of mode selection will help prevent unintended data loss when the mode is potentially destructive (e.g., DELETE). For destructive modes, it may help if the mode indication is implemented as some sort of distinctive change in the appearance of the cursor, since the cursor is probably the one display feature most surely seen by a user.

Reference: BB 4.3.4; MS 5.15.7.5.a; Foley and Wallace, 1974.

See also: 3.4-4, 3.4-5, 4.1-5, 4.4-10, 6.0-4, 6.5-13.

-9 o Indicating Option Selection -9

When previously selected options are still operative, display those options either automatically or on user request.

Comment: A displayed "audit trail" of option selections will help a novice user learn transaction sequences, and may help any user deal with complex transactions.

Reference: EG 3.4.

See also: 4.4-10, 4.4-18.

-10 o Indicating Item Selection -10

When a user selects a displayed item in order to perform some operation on it, highlight that item on the display.

Comment: This practice will provide a routine natural feedback that item selection has been accomplished, and will provide a continuing reminder to the user of just what selection has been made.

Comment: Highlighting might be accomplished in different ways. Reverse video is commonly employed for this purpose. For a selection among displayed options, the selected option might be brightened, or the non-selected options might be dimmed.

Reference: EG 2.1.1, 3.1, 3.1.1; MS 5.15.5.6.

See also: 1.1-5, 3.1.3-9, 3.4-6.

-11 • Feedback for User Interrupt -11

Following user interrupt of data processing, display an advisory message assuring the user that the system has returned to its previous status.

Reference: BB 4.7.

See also: 3.3-6.

When an error or other unexpected event prevents routine transaction processing it is important to inform the user.

-1 • Informative Error Messages -1

When the computer detects an entry error, display an error message to the user stating what is wrong and what can be done about it.

Example: (Good) Code format not recognized; enter two letters, then three digits.

(Bad) Invalid input.

Comment: Users should not have to search through reference information to translate error messages.

Comment: Error messages can be regarded as the most important form of system documentation. Well designed error messages will give help to users automatically, at the point where help is most needed.

Reference: BB 5.2.2, 5.3.2, 5.3.8; EG 3.3.1; MS 5.15.5.7, 5.15.7.5; PR 4.12.1; Dean, 1982; Limanowski, 1983; Magers, 1983.

-2 o Specific Error Messages -2

Make the wording of error messages as specific as possible.

Example: (Good) No record for Loan 6342; check number.

(Bad) No record for inquiry.

Comment: Specificity will require computer analysis of data processing transactions in context.

Reference: BB 5.3.7; MS 5.15.7.6, 5.15.7.8; PR 4.12.5.1.

-3 o Task-Oriented Error Messages -3

Adopt wording for error messages which is appropriate to a user's task and level of knowledge.

Example: (Good) Contract number not recognized; check the file and enter a current number.

(Bad) Entry blocked. Status Flag 4.

Comment: Error messages that can be understood only by experienced interface designers and programmers may have no value for ordinary users.

Reference: BB 5.3.5; EG 3.3.7; MS 5.15.7.6.

See also: 2.0-11, 4.0-17.

-4 • Advisory Error Messages -4

If a data entry or (more often) a control entry must be made from a small set of alternatives, an error message that is displayed in response to a wrong entry should indicate the correct alternatives.

See also: 3.2-5, 4.4-1.

-5 • Brief Error Messages -5

Make error messages brief but informative.

Example: (Good) Entry must be a number.

(Bad) Alphabetic entries are not acceptable because this entry will be processed automatically.

Comment: Often a user will recognize that an error has been made, and the message will serve merely as a confirming reminder. In such instances, short error messages will be scanned and recognized more quickly.

Comment: For a user who is truly puzzled, and who needs more information than a short error message can provide, auxiliary HELP can be provided either on-line or by reference to system documentation.

Comment: If an on-line HELP explanation is not available, a user may have to refer to system documentation for a coded listing of possible errors. Under those circumstances, some designers display each error message with an identifying code, to facilitate rapid reference to documentation. That practice might help experienced users, who would gradually come to recognize the codes.

Reference: BB 5.3.4; EG 3.1.3, 3.3, 3.3.7; PR 2.2, 4.1.2.2.

See also: 2.1.1-10, 2.1.1-11, 4.4-19.

-6 • Neutral Wording for Error Messages -6

Adopt neutral wording for error messages; do not imply blame to the user, or personalize the computer, or attempt to make a message humorous.

Example: (Good) Entry must be a number.

(Bad) Illegal entry.

(Bad) I need some digits.

(Bad) Don't be dumber, use a number.

Comment: Error messages should reflect a consistent view that the computer is a tool, with certain limitations that a user must take into account in order to make the tool work properly. If error messages reflect an attitude that the computer (or its programmer) imposes rules, or establishes "legality", the user may feel resentful. If error messages reflect personalization of the computer, as if it were a friendly colleague, a naive user may be misled to expect human abilities the machine does not actually possess. If error messages are worded humorously, any joke will surely wear thin with repetition, and come to seem an intrusion on a user's concern with efficient task performance.

Comment: The same considerations apply for the wording of computer-generated prompts and other instructional material.

Reference: BB 5.5.3; EG 3.3.8, 5.3; MS 5.15.7.6; PR 2.2.

-7 • Multilevel Error Messages -7

Following the output of a simple error message, permit users to request a more detailed explanation of the error.

Comment: A more complete discussion of each error could be made available on-line, perhaps at several levels of increasing detail, supplemented by reference to off-line system documentation if necessary. Successively deeper levels of explanation might then be provided in response to repeated user requests for HELP.

Reference: BB 1.6, 5.4; EG 3.3.

See also: 4.4-23.

-8 • Multiple Error Messages -8

When multiple errors are detected in a combined user entry, notify the user, even though complete messages for all errors cannot be displayed together.

Example: DATE should be numeric. [+ 2 other errors]

Comment: The computer should place the cursor in the data field referred to by the displayed error message, with other error fields highlighted in some way (e.g., by reverse video). There should also be some means for the user to request sequential display of the other error messages if needed.

Reference: PR 4.12.3.

See also: 4.3-17.

-9 o Indicating Repeated Errors -9

If a user repeats an entry error, there should be some noticeable change in the displayed error message.

Example: Some designers go to the extent of providing two versions of each error message, which alternate on the display in response to repeated errors.

Comment: If an error message is repeated identically, so that displayed feedback seems unchanged, the user may be uncertain whether the computer has processed the revised entry. A simple expedient might be to display the same verbal message but with changing annotation, perhaps marked with one asterisk or two.

-10 • Non-Disruptive Error Messages -10

The computer should display an error message only after a user has completed an entry.

Example: An error message should not be generated as wrong data are keyed, but only after an explicit ENTER action has been taken.

Comment: In general, the display of error messages should be timed so as to minimize disruption of the user's thought process and task performance.

Reference: EG 7.1.

See also: 1.7-3, 1.7-7.

-11 • Appropriate Response Time for Error Messages -11

Display an error message approximately 2-4 seconds after the user entry in which the error is detected.

Exception: For type-ahead systems with experienced users, error messages should be displayed as quickly as possible.

Comment: Longer delays in error feedback may cause user uncertainty or confusion. Longer delays may also cause frustration if the user is already aware of the error, which is often the case.

Comment: Shorter delays in error feedback can pose problems of a different sort. An error message following immediately upon a user entry can be disconcerting. Immediate error feedback can also be irritating. User expectations are conditioned by human conversation, where an immediate contradiction is considered rude, and where a polite listener will think for a few moments before saying that you are wrong.

Comment: If error messages take somewhat longer to appear than the routine computer response, then that additional delay may help condition the user to expect an error message and pay attention to it.

Reference: EG Table 2.

See also: 3.0-15.

Many aspects of interface design, and user guidance, can be assessed and improved by recording user performance.

-1 • User Performance Measurement -1

In applications where skilled user performance is critical to system operation, provide automatic computer recording and assessment of appropriate user abilities.

Comment: Recording individual performance may be constrained by other considerations, as noted elsewhere in this section.

See also: 4.4-27.

-2 • Notifying Users -2

Inform users of any records kept of individual performance.

Comment: Informing the user concerning the nature and purpose of performance records is required by ethical principle, and in some situations may be required by law.

Comment: Recording individual performance is a practice that is potentially subject to abuse, and requires careful scrutiny to ensure essential protection of user privacy. Designers must conform to whatever legal (or otherwise agreed) restrictions may be imposed in this regard.

-26 o Flexible Training -26

Anticipate the needs of different users and offer different levels of training for on-line job support.

Example: In systems supporting different user jobs, on-line instruction might describe the procedures for each different data handling task.

Example: Instruction on keyboard use and lightpen selection of menu options might be provided for novice users, while a tutorial on command language might be provided for more experienced users.

See also: 3.0-3, 3.1-1, 3.1.3-33, 3.1.5-4, 4.0-22.

-27 o Adaptive Training -27

In applications where a user must learn complex tasks, design computer-mediated training to adapt automatically to current user abilities.

Comment: Adaptive training will require some means for computer assessment of appropriate components of user performance.

See also: 4.0-22, 4.5-1.

-23 o Multilevel HELP -23

When an initial HELP display provides only summary information, provide more detailed explanations in response to repeated user requests for HELP.

Comment: It is necessarily a matter of judgment just what information should be provided in response to a HELP request. Designing the HELP function to provide different levels of increasing detail permits users to exercise some judgment themselves as to just how much information they want.

Reference: BB 5.4, 6.3.

See also: 4.3-7.

-24 o Browsing HELP -24

Permit novice users to browse through on-line HELP displays, just as they would through a printed manual, to gain familiarity with system functions and operating procedures.

Reference: Cohill and Williges, 1982.

-25 • On-Line Training -25

Where appropriate, provide an on-line training capability to introduce new users to system capabilities and to permit simulated "hands on" experience in data handling tasks.

Comment: On-line simulation, using the same hardware, user interface software and data processing logic as for the real job, can prove an efficient means of user training. Care must be taken, however, to separate the processing of simulated data from actual system operations.

Reference: BB 6.4.

See also: 6.2-2, 6.3-4, 6.5-4.

-20 o Standard Action to Request HELP -20

Provide a simple, standard action that is always available to request HELP.

Example: HELP might be requested by an appropriately labeled function key, or perhaps by keying a question mark into a displayed entry area.

Comment: The user should be able to request HELP at any point in a transaction sequence. The procedure should always be the same, whether the user wants an explanation of a particular data entry, a displayed data item, or a command option.

Reference: BB 4.4.3; Keister and Galaway, 1983.

-21 o Task-Oriented HELP -21

Tailor the response to a HELP request to task context and the current transaction.

Example: If a data entry error has just been made, HELP should display information concerning entry requirements for that particular data item.

Example: If an error in command entry has just been made, HELP should display information concerning that command, its function, its proper structure and wording, required and optional parameters, etc.

Reference: BB 6.3; Magers, 1983.

-22 o Clarifying HELP Requests -22

When a request for HELP is ambiguous in context, the computer should initiate a dialogue in which the user can specify what data, message or command requires explanation.

Example: The computer might ask a user to point at a displayed item about which HELP is requested.

-17 • Definition of Display Codes -17

When codes are assigned special meaning in a particular display, display a definition of those codes.

Comment: This practice will aid user assimilation of information, especially for display codes that are not already familiar.

Reference: BB 7.6.1.

See also: 2.4-5.

-18 • Record of Past Transactions -18

Where appropriate, permit users to request a displayed record of past transactions in order to review prior actions.

Reference: EG 4.2.7.

See also: 3.4-3, 4.5-3.

-19 • HELP -19

In addition to explicit aids (labels, prompts, advisory messages) and implicit aids (cueing), permit users to obtain further on-line guidance by requesting HELP.

Comment: It is difficult for an interface designer to anticipate the degree of prompting that may be required to guide all users. Moreover, even when prompting needs are known, it may be difficult to fit all needed guidance information on a display page. A supplementary HELP display can be provided to deal with such situations.

Reference: BB 4.4.3; 6.3; MS 5.15.7.5; PR 3.3.15.

-14 o Index of Data -14

In applications where the user can choose what stored data to display, provide an on-line data index to guide user selection.

Comment: The data index should indicate file names, objects, properties, and other aspects of file structure that might be used to access different categories of data. It may help to allow users to specify what they require in this index. Some users might wish information on file size, currency (time of latest update), etc. Other users might wish to add a short description of each data file to remind themselves of its contents.

Reference: BB 6.2.

See also: 3.1.6-3.

-15 o Index of Commands -15

In applications where a user can employ command entry, provide an on-line command index to guide user selection and composition of commands.

Comment: Such a command index may help the user to phrase a particular command, and will also be generally helpful as a reference for discovering related commands and learning the overall command language.

Reference: BB 6.2, 6.3; Magers, 1983.

-16 o Dictionary of Abbreviations -16

Provide a complete dictionary of abbreviations used for data entry, data display, and command entry, for on-line user reference and in design documentation.

Comment: In applications where users can create their own abbreviations, as in the naming of command "macros", it will be helpful to provide aids for users to create their own individual on-line dictionaries.

Reference: BB 6.5; MS 5.15.6.5.

See also: 2.1.1-28.

-12 • Consistent Cursor Positioning -12

As the last step in generating a display output, the computer should automatically position the cursor so that it appears in a consistent display location for each type of transaction.

Example: For data entry displays, the cursor should be placed initially at the first data field, or else at the first wrong entry if an error has been detected. In other displays, the cursor should be placed at a consistent HOME position, or at the first control option for menu selection, or else in a general command entry area, depending upon the type of display.

Comment: Consistent cursor positioning will provide an implicit cue for user guidance.

Reference: EG 4.2.3; MS 5.15.2.1.8.3; PR 3.3.3.

See also: 1.1-20, 1.1-21, 1.4-26, 3.2-6, 3.2-7, 4.3-13.

-13 • On-Line System Guidance -13

Provide reference material describing system capabilities and procedures available to users for on-line display.

Comment: Many systems are not utilized effectively because users do not fully understand system capabilities. On-line access to a description of system structure, components and options will aid user understanding.

Comment: On-line guidance can supplement or in some instances substitute for off-line training. An investment in designing user aids may be repaid by reduced costs of formal training as well as by improved operational performance.

Reference: BB 6.1, 6.2; Limanowski, 1983.

-10 • Displayed Context -10

When the results of a user entry depend upon context established by previous entries, display some indication of that context to the user.

Example: When the effects of user entries are contingent upon different operational modes, indicate the current mode.

Example: If the user is editing a data file, display both the file name and an indication of EDIT mode.

Reference: EG 4.2.1; MS 5.15.4.1.5, 5.15.7.5.

See also: 2.7-5, 2.7-6, 3.0-9, 3.1.4-5, 3.1.4-9, 3.4-1, 3.4-4, 3.4-5, 4.1-5, 4.2-8.

-11 • Cues for Prompting Data Entry -11

Provide cues for data entry by formatting data fields consistently and distinctively.

Example: A colon might be used consistently to indicate that an entry can be made, followed by an underscored data field to indicate item size, such as

Enter part code: _ _ _-_ _

or perhaps just simply

PART CODE: _ _ _-_ _

Comment: Consistent use of prompting cues can sometimes provide sufficient guidance to eliminate the need for more explicit advisory messages.

Reference: BB 2.1.5; EG 6.3.1.

See also: 1.4-9 thru -11, 1.4-16.

-7 • Prompting Entries -7

Provide advisory messages and other prompts to guide users in entering required data and/or command parameters.

Comment: Prompting in advance of data/command entry will help reduce errors, particularly for inexperienced users.

Comment: If a default value has been defined for null entry, that value should be included in the prompting information.

Reference: EG 4.2.2, 4.2.4; MS 5.15.4.1.4, 5.15.7.5; PR 4.9.2; Foley and Wallace, 1974.

See also: 1.0-23, 1.8-3, 3.1.5-9, 3.2-11, 3.5-5.

-8 o Standard Display Location for Prompting -8

Display prompts for data/command entry in a standard location, next to the command entry area at the bottom of the display.

See also: 4.0-6.

-9 o User-Requested Prompts -9

When users vary in experience, which is often the case, provide prompting as an optional guidance feature that can be selected by novice users but can be omitted by experienced users.

Comment: Flexibility in prompting can also be provided by multi-level HELP options, so that additional guidance information can be obtained if the simple prompt is not adequate.

See also: 3.1.5-9, 3.1.5-11, 4.4-23.

-4 • Hierarchic Menus -4

When hierarchic menus must be used, organize and label them to guide users within the hierarchic structure.

Comment: Users will learn menus more quickly if a map of the menu structure is provided as HELP.

Reference: Billingsley, 1982.

See also: 3.1.3-22, 3.1.3-23, 3.1.3-28.

-5 • Guidance for Sequence Control -5

At every point in a transaction sequence, provide guidance telling the user how to continue.

Example: (Good) Data are current through March 1983.
Press STEP key to continue.

(Bad) Data are current through March 1983.

Reference: BB 4.2; EG 3.1.2; PR 2.2.

See also: 3.0-4, 3.1.3-14, 3.2-12.

-6 o Transaction-Specific Option Display -6

If there are control options that are specifically appropriate to the current transaction, indicate those options on the display.

Exception: Treat control options that are generally available at every step in a transaction sequence (such as PRINT, perhaps) as implicit options that need not be included in a display of specific current options.

Comment: Usually space can be found on a working display to remind users of several specific control options that are appropriate to the current transaction. A list of all available options, however, may well exceed display capacity. A user may be expected to remember general options, once they have been learned, without their specific inclusion in a display of guidance information. Perhaps the best design approach is to implement general options on appropriately labeled function keys, which will aid user learning and provide a continuing reminder of their availability.

Specific task-oriented guidance should be provided throughout the transaction sequences that comprise a user's job.

-1 • Guidance Information Always Available -1

Ensure that specific user guidance information is available for display at any point in a transaction sequence.

Comment: Do not require a user to remember information not currently displayed. The user should not have to remember what actions are available, or what action to take next. Human memory is unreliable, and without guidance the user can be expected to make errors.

Reference: BB 4.3.6; EG 3.4.4; MS 5.15.4.1.7.

See also: 2.0-3, 2.2-1, 2.6-1, 3.1.3-15, 3.1.3-16, 3.2-4, 3.2-5.

-2 • General List of Control Options -2

Provide a general list (or menu) of control options that is always available to serve as a "home base" or consistent starting point to begin a transaction sequence.

Reference: BB 4.1, 4.4.5.

See also: 3.1.5-10, 3.2-2, 3.2-3.

-3 • Logical Menu Structure -3

Display menu options in logical groups to aid user learning and selection among displayed alternatives.

Comment: It may be necessary to test proposed menus to determine just what structural relationships will seem logical to their intended users.

See also: 2.1.1-18, 3.1.3-20, 3.2-3.

-17 • Alarm Coding -17

For conditions requiring (or implying the need for) special user attention, code the alarms (or warning messages) distinctively.

Example: Alarm messages might be marked with a blinking symbol and/or displayed in red, and be accompanied by an auditory signal. Warnings and error messages might be marked with a different special symbol and/or displayed in yellow.

Comment: This practice will help ensure appropriate attention, even when the user is busy at routine tasks.

Reference: BB 1.1.2, 7.7.2, 7.7.3; EG 2.1.3; MS 5.15.3.3.2.

See also: 2.4-11, 2.4-29, 2.4-31, 2.4-36, 3.6-2, 6.0-8, 6.5-17.

-15 • Cautionary Messages -15

When a user entry is recognized as doubtful, when analyzed by defined data/command validation logic, display a cautionary message asking the user to confirm that entry.

Example: Blood pH of 6.6 is outside the normal range; confirm or change entry.

Comment: Feedback to the user can be worded to deal with a range of intermediate categories between a seemingly correct entry and an outright error.

Reference: MS 5.15.7.2.

See also: 3.5-8, 3.5-11, 4.3-16.

-16 • User Confirmation of Destructive Entries -16

Require the user to take some explicit action to confirm a potentially destructive data/command entry before the computer will execute it.

Comment: A requirement to take an explicit CONFIRM action will direct user attention to questionable entries and help the user avoid the consequences of thoughtless errors.

Comment: What constitutes "potentially destructive" requires definition in the context of each system application.

Reference: BB 5.6; EG 4.2.8; Foley and Wallace, 1974.

See also: 3.5-7, 4.3-15, 6.0-8, 6.3-20, 6.5-14, 6.5-18.

-12 • Documenting Error Messages -12

As a supplement to on-line guidance, include in the system documentation a listing and explanation of all error messages.

Comment: Developing good error messages may require review by both designers and users. Documentation of the complete set of error messages will facilitate such review.

Comment: Documentation of error messages will permit users to reference particular messages for fuller explanation, and to review all messages as a means of understanding data processing requirements and limitations.

Reference: BB 5.3.1, 5.4, 5.8.

-13 • Cursor Placement Following Error -13

In addition to providing an error message, mark the location of a detected error by positioning the cursor at that point on the display, i.e., at that data field or command word.

Comment: Displaying the cursor at a non-routine position will help emphasize that an error has occurred, and direct the user's attention to the faulty entry.

Reference: BB 5.2.5; PR 4.12.1.

See also: 4.4-12.

-14 • User Editing of Entry Errors -14

Following error detection, prompt the user to re-enter only the portion of a data/command entry that is not correct.

Comment: The user should not have to re-key an entire command string or data set just to correct one wrong item.

Reference: BB 5.2.1; EG 4.2.3; MS 5.15.7.1.

See also: 3.1.5-19, 3.5-3, 6.0-6, 6.3-12.

-3 • Transaction Records -3

In applications where it is warranted, the computer should maintain records of user transactions.

Comment: Transaction recording might be made optional, under control of a system manager.

Comment: Record keeping might include duration, sequencing and frequency of different transactions. Such transaction records will aid task analysis, particularly in developing systems where data handling requirements are not yet fully defined.

Comment: Buffered store of current transaction sequences may be required for user guidance.

See also: 4.4-18.

-4 • Data Access Records -4

In applications where it is warranted, the computer should maintain records of data access, i.e., which data files, categories, items have been called out for display.

Comment: Records of data use may help software designers improve file structure, reduce data access time, and manage multiple use of shared data files.

Comment: Data access records may also be required for purposes of data protection/security.

See also: 6.2-8.

-5 • Program Use Records -5

In applications where it is warranted, the computer should maintain records of use for different portions of applications software.

Comment: In many cases "program calls" can be derived from transaction records rather than measured directly.

Comment: Records of software use may not affect user interface design directly, but can help detect and correct programming inefficiencies and improve system response, particularly during early stages of system development.

-6 • Error Records -6

Provide a capability for recording user errors.

Comment: Error recording might be done continuously, or by periodic sampling, under the control of system supervisors.

Comment: Error records can be used to indicate supplemental instruction needed by different users, if individual user errors are identified. In that case, ethical considerations (and in some instances legal considerations) dictate that users should be informed that such records will be kept.

Comment: Error records can be used to indicate whether particular transactions are giving trouble to many users, in which case design improvements to the user interface may be needed, including changes to user guidance.

Comment: For an individual user, the computer might be programmed to generate a special on-line HELP sequence to guide the correction of repeated errors.

Reference: BB 5.7.

See also: 4.5-2, 4.6-1.

-7 • HELP Records -7

Provide a capability for recording user requests for HELP.

Exception: There is probably no need to record user browsing of HELP messages, if such a capability is provided.

Comment: HELP records can be used to detect deficiencies in in early system development, and can be used to improve user guidance in later system operation. In effect, user requests for HELP might be regarded as a possible symptom of poor interface design. If HELP requests are frequent for a particular transaction, then some design improvement may be needed, in procedures, or prompting for user guidance, or both.

See also: 4.4-19, 4.4-24.

Changes to the software design of user guidance functions may be needed to meet changing operational requirements.

-1 • Flexible Design for User Guidance -1

When user guidance requirements may change, which is often the case, provide some means for users (or a system administrator) to make necessary changes to user guidance functions.

Comment: User guidance functions that may need to be changed include those represented in these guidelines, namely, changes in status information, routine and error feedback, job aids and user records.

Comment: Many of the preceding guidelines in this section imply a need for design flexibility. Much of that needed flexibility can be provided in initial interface design. Some guidelines, however, suggest a possible need for subsequent design change, and those guidelines are cited below.

Comment: In some applications, it may prove helpful to allow individual users to reword and/or add their own notes to on-line guidance material, just as they might annotate paper documentation.

Reference: Limanowski, 1983.

See also: 4.3-12, 4.4-3, 4.4-16, 4.5-3, 4.5-5 thru -7.

-2 • Notifying Users of Design Changes -2

When changes are made to interface design, including changes to user guidance functions and on-line documentation, inform users of those changes.

Comment: An on-line "message board" appearing at LOG-ON may suffice to notify users of current changes. But more extensive measures may be needed, including corresponding changes to user guidance information, e.g., prompts, error messages, HELP displays, etc.

Reference: Limanowski, 1983.

SECTION 5

DATA TRANSMISSION

Data transmission refers to message exchange among the various users of a system, and also to message exchange with other systems. Preceding sections of this report have dealt with the basic functions of using on-line information systems -- entering data into a computer, displaying data from a computer, controlling the sequence of input-output transactions, with guidance for users throughout the process. What other functions can a computer serve? One area that increasingly demands our attention is the use of computers for communication, i.e., to mediate the transmission of data from one person to another.

In considering data transmission functions, we must adopt a broad perspective. Data that are transmitted via computer may include words and pictures as well as numbers. And the procedures for data transmission may take somewhat different forms for different system applications.

In some applications, computer-mediated data transmission may be a discrete, task-defined activity. Perhaps a system used for planning/scheduling is later used to generate and transmit orders to implement a plan. In such a system, a user can "shift gears", first creating a plan and then transmitting that plan.

In other applications, data transmission may be a continuing, intermittent activity mixed with other tasks. As an example, air traffic controllers might use their computer facilities to exchange information (and to hand over responsibility) while they are performing other essential flight monitoring tasks.

An even broader requirement for computer-based message handling can be seen in systems whose explicit, primary purpose is to support communication. In such applications, computer-mediated data transmission is sometimes called "electronic mail". In conjunction with other new technology, the current development of electronic mail has brought forecasts of a "wired society" in which we become ever more dependent on computers for communication (Martin, 1978).

Effective communication is of critical importance in systems where information handling requires coordination among groups of people. This will be true whether communication is mediated by computer or by other means. Computer-based message handling may offer a potential means of improving communication efficiency, but careful design of the user interface will be needed to realize that potential.

In the interests of efficiency, much data transmission among computers is designed to be automatic, representing a programmed message exchange between one computer and another, with no direct user involvement. If a user does not participate in data transmission, then there is of course no need to include data transmission functions in user interface design. Only when the users themselves are involved in data transmission transactions will interface design guidelines be needed.

For the users of computer systems, data transmission can impose an extra dimension of complexity. A user not only must keep track of transactions with the computer, but also must initiate and monitor data exchange with other people. Data exchange may be with other users of the same system, and/or with the users of other systems. Data exchange may be immediate, with other currently active users. Or data exchange may be deferred until other users come on-line to receive their messages, in which case the process may extend over a period of time.

Computer users will need extra information to control data transmission, perhaps including status information about other systems, and the communication links with other systems. Users will need feedback when sending or receiving data. Users may need special computer assistance in composing, storing and retrieving messages, as well as in actual data transmission.

The general objectives of user interface design in other functional areas are equally valid for data transmission functions. Procedures for data transmission should be consistent in themselves, and consistent with procedures for data entry and display. Interface design should minimize effort and memory load on the user, and permit flexibility in user control of data transmission.

When data transmission functions can be designed as an integral part of an information system, there is a clear opportunity for the interface designer to ensure compatibility of procedures. For example, if the system provides procedures for text editing, file storage, and retrieval in support of so-called "word processing" functions, then those same procedures should serve to edit, store, and retrieve messages. Users will not have to learn a different set of commands, or select from a different assortment of menu options.

In some current applications, however, data transmission functions have been grafted onto an existing system. There is a practical advantage in buying a separate package of message handling software for use in conjunction with an already existing system. The message handling functions do not have to be designed from scratch, and they can often be added without any fundamental redesign of the existing system capabilities.

There is a potentially serious disadvantage, however, in trying to combine separately designed software packages: they will almost surely look different to a user, unless care is taken to design an integrated user interface. Differences in user interface logic can sometimes be "papered over" by the software designer, perhaps by providing a menu overlay that effectively conceals inconsistencies of control logic (Goodwin, 1983; 1984). How well the user interface design has integrated the disparate software packages should then be evaluated in operational use (Tammaro, 1983).

Whether the user interface to data transmission functions can be designed from scratch, or is designed separately and then must be integrated with other system functions, some guidelines may be needed to help the interface designer. Recent studies of computer-based message handling have been chiefly concerned with determining the functional capabilities required in data transmission (cf., Goodwin, 1980). There is already evidence, however, that the practical use of data transmission functions can be limited by deficiencies in user interface design (Goodwin, 1982).

Until the recent guidelines compilation by Smith and Aucella (1983), no recommendations were available for interface design of data transmission functions. The revised guidelines proposed here still offer little reference to supporting data or independent recommendations. Critical review is much needed in this functional area.

- Data transmission refers to message exchange among system users, and also to message exchange with other systems.
- Transmitted messages might include free and formatted text, as well as data forms, file displays, etc.
- Users may need to specify data source and destination when sending messages.
- Users may need to specify data source and (device) destination when receiving messages.
- Users may need to control the sequence, content, format, routing, timing, etc., of data transmissions.
- Feedback of information related to data transmission will be needed to permit users to exercise effective control.
- Queuing of messages will be needed to permit transmission control by users, both for sending and receiving.
- Users may need computer aids to keep track of data transmission.
- Changes to the software design of data transmission functions may be needed to meet changing operational requirements.

Objectives:

Consistency of data transmission
Minimal user actions
Minimal memory load on user
Compatibility with other data handling
Flexibility for user control of data transmission

Data transmission refers to message exchange among system users, and also message exchange with other systems.

-1 • Consistent Procedures -1

Procedures for data transmission, i.e., for sending data or for receiving data, should be consistent from one transaction to another.

-2 • Minimal User Actions -2

Design of data transmission procedures should minimize user actions required.

Example: Interface software might provide automatic formatting of messages derived from data already stored in the computer.

Example: Software might provide automatic reformatting of stored data for transmission, where format change is required.

Example: Automatic queuing of outgoing messages pending completion of transmission, and incoming messages pending review and disposition.

-3 • Minimal Memory Load on User -3

Design of data transmission procedures should minimize memory load on the user.

Example: Interface software might provide automatic insertion into messages of standard header information, distribution lists, etc.

Example: Software might provide automatic record keeping, message logging, status displays, etc.

-4 • Message Composition Compatible with Data Entry -4

Procedures for sending data, i.e., for composing messages, should be compatible with procedures for general data entry.

Comment: A user should not have to learn procedures for entering messages that are different from more general data entry, particularly if those procedures might involve conflicting habits.

See also: Section 1.

-5 o Message Viewing Compatible with Data Display -5

Procedures for receiving data, i.e., for review of incoming messages, should be compatible with procedures for general data display.

Comment: A user should not have to learn procedures for reviewing messages that are different from more general data display, particularly if those procedures might involve conflicting habits.

See also: Section 2.

-6 • Flexible User Control -6

Permit flexible user control of data transmission, so that the user can decide what data should be transmitted, when, and where.

Exception: In monitoring and control applications, data processing and transmission must be event-driven.

Comment: Where routine data transmission is required, that should not interfere with a user's other information handling activities.

-7 • Control by Explicit User Action -7

Design the data transmission procedures to permit both sending and receiving messages by explicit user action.

Comment: Automatic message generation and receipt will be helpful in many applications, but in such cases the user should be able to monitor transmissions, and should be able to participate by establishing, reviewing and/or changing the computer logic controlling automatic data transmission.

See also: 1.0-8, 3.0-5, 4.0-2, 6.0-3.

Transmitted messages might include free and formatted text, as well as data forms, file displays, etc.

-1 • User-Designed Formats -1

Where required message formats will vary unpredictably, permit users to compose and transmit messages with a format of their own design.

Comment: Establishing new formats, particularly if automatic data validation must be specified for defined fields, may require special tools, and so be a task properly delegated to a skilled system administrator and not to all system users.

-2 • Automatic Text Formatting -2

When transmitted text must be formatted in a particular way, format control should be automatic, with no extra attention required from the user.

Example: Defined message formats might be filled automatically from stored data.

Example: Header/paging formats might be inserted automatically in document transmission.

Comment: A user should not have to transpose data for transmission in a form different than that used originally for data entry.

See also: 1.3-21, 5.0-2, 5.4-3.

-3 • Unformatted Text -3

Permit users to compose and transmit messages as unformatted text.

Comment: Allowing users to create arbitrary text messages (sometimes called "chatter") will let users deal flexibly with a variety of communication needs not anticipated by system designers.

-4 • Data Forms -4

In transmission of data forms, permit users to enter, review and change data using an organized display with field labels, rather than requiring them to deal with an unlabeled string of items.

Comment: User composition and review of unlabeled data strings, especially those requiring delimiters to mark items, will be prone to error. If such data strings are needed, they should be generated automatically from data entered in a form-filling dialogue.

Comment: Transmission of data from one computer to another will often be more economical if field labels and other display formatting features are omitted. In such cases, a format code should be included with the message, so that forms filled by the sender can be re-created in a display useful to the receiver.

See also: 5.0-4, 5.0-5.

-5 o Tables and Graphics -5

In transmission of tabular or graphic data, permit users to enter, review and change data in customary formats, regardless of what the computer-imposed format is for actual transmission.

See also: 5.0-4, 5.0-5, 5.1-4.

-6 • Message Highlighting -6

When it will help message handling, annotate transmitted data with appropriate highlighting to emphasize alarm/alert conditions, priority indicators, or other significant second-order information.

Comment: Second-order information, i.e., data about data, will often aid processing and interpretation of messages. Such annotation can be automatic (e.g., a computer-generated date-time stamp to indicate currency) and/or added by the sender or receiver (e.g., attention arrows).

See also: Section 2.7.

Changes to the software design of data transmission functions may be needed to meet changing operational requirements.

-1 • Flexible Design for Data Transmission -1

When data transmission requirements may change, which is often the case, provide some means for users (or a system administrator) to make necessary changes to transmission functions.

Comment: Data transmission functions that may need to be changed include those represented in these guidelines, namely, changes in the types of transmitted data, sending, receiving, and transmission control procedures, feedback, queuing, and record keeping.

Comment: Many of the preceding guidelines in this section imply a need for design flexibility. Much of that needed flexibility can be provided in initial interface design. Some guidelines, however, suggest a possible need for subsequent design change, and those guidelines are cited below.

See also: 5.0-6, 5.1-1, 5.2-2, 5.3-1, 5.4-2, 5.5-4.

Users may need computer aids to keep track of data transmission.

-1 • Automatic Record Keeping -1

When a log of data transmissions is required, maintain that log automatically, based on user specification of message types and record formats.

Comment: The objective here is to minimize routine "housekeeping" chores for the user. Once a user has specified the appropriate logging format, i.e., what elements of each message should be recorded, computer aids should generate the log automatically, either for all messages, or for specified categories of messages, or for particular messages identified by the user.

Comment: The same kind of aids should be available for maintaining a journal of data transmissions, in applications where a full copy of each message is required.

See also: 5.0-2, 5.0-3.

-7 • User Review of Messages Received -7

Provide convenient means for user review of received messages in a queue, without necessarily requiring any disposition action (i.e., without removal from the queue).

Exception: In some applications, user review of critical messages may be accompanied by a requirement for further disposition actions.

Comment: Rapid review of queued messages will permit a user to exercise judgment as to which require immediate attention, and/or which can be dealt with easily. Other messages may be left in the queue for more leisurely disposition later.

-4 • Queuing Messages Received -4

Unless otherwise specified by a user, the computer should route all incoming messages automatically to a queue, pending subsequent review and disposition by the user.

Comment: Some computer buffering of received data transmissions will be required in any case to deal with near simultaneous receipt of multiple messages. This guideline recommends that the buffer queue for incoming transmissions be enlarged as necessary to permit indefinite retention of messages. Any queue will have limits, of course, and the user should be warned before those limits are exceeded.

See also: 5.0-6, 5.0-7.

-5 • Non-Disruptive Message Receipt -5

Permit receipt of messages without interfering with a user's ongoing task.

Comment: An incoming message should not preempt a user's display. Instead, the computer might indicate message receipt to a user by an advisory notice in a portion of the display reserved for that purpose.

Comment: Review and disposition of received messages, like other transactions, should generally require explicit actions by the user. When an incoming message implies an urgent need for user attention, notify the user.

Reference: EG 7.1.

See also: 5.0-7, 5.3-2, 5.6-6, 6.4-4.

-6 o Indicating Priority for Messages Received -6

When received messages are queued, notify users of message priority and/or other information indicating urgency of user action.

Comment: If incoming messages are queued so as not to disrupt current user tasks, it is important that users be advised when an interruption may in fact be necessary.

See also: 5.2-4, 5.3-3, 5.5-3, 5.6-5.

Queuing of messages will be needed to permit transmission control by users, both for sending and receiving.

-1 • Automatic Queuing -1

Provide automatic message queuing to reduce the need for user involvement in the routine mechanics of data transmission.

Comment: Specific requirements will vary with the application, but some queuing should be provided.

-2 • Deferring Message Transmission -2

Permit users to defer the transmission of outgoing messages, to be released by later action.

Comment: A user may wish to defer data transmission until a batch of related messages has been prepared, or perhaps until a specified date-time for release.

-3 o Queuing Failed Transmissions -3

The computer should queue outgoing messages automatically in the event of transmission failure.

Comment: In the event of transmission failure, automatic queuing and retransmission of outgoing messages will reduce load on the user. If transmission fails in repeated attempts, however, then user intervention may be required, and some notification of that problem should be given to the user.

See also: 5.5-2.

-4 • User Specification of Feedback -4

Permit users to specify what feedback should be provided for routine message transmission, and also to request specific feedback for particular messages.

Comment: Users may wish to specify minimal feedback, or perhaps none at all, as the automatic notification of routine messages. On the other hand, users may wish to request more specific feedback for transmission of critical messages, as an electronic version of registered mail.

Feedback of information related to data transmission will be needed to permit users to exercise effective control.

-1 • Automatic Feedback -1

Provide automatic feedback for data transmission as necessary to permit effective user participation in message handling, confirming that messages are sent, announcing when messages are received, indicating message priority, etc.

Comment: Specific requirements will vary with the application, but some feedback should be provided.

-2 • Feedback for Message Sent -2

Permit users sending messages to request feedback to confirm that a message has been sent and/or received, or to indicate if there has been a transmission failure.

Comment: In some applications, users may require notification only of exceptional circumstances, as in the event of transmission failure after repeated attempts.

Comment: For the computer equivalent of registered mail, it might be helpful in some applications to provide the sender some explicit confirmation that a message has been safely received, or even to notify the sender when a recipient has actually accessed the message.

See also: 5.6-2.

-3 • Information About Messages Received -3

Permit users receiving messages to request information about the type, source and priority of incoming data transmissions.

Comment: In some applications, a user may need notification only of urgent messages, and rely on periodic review to deal with routine messages.

See also: 5.6-6.

-6 • User Review of Transmitted Data -6

When computer aids are provided for selection, routing and initiation of data transmission, users should have the option of reviewing and changing automated message handling logic, in general and for selected messages prior to transmission.

Comment: In applications where message review is critical (perhaps for purposes of security), users might be required to confirm data release for transmission.

See also: 5.4-4, 5.4-5, 6.4-2.

-3 • Automatic Message Formatting -3

When data must be transmitted in a particular format, as in data forms or formatted text, computer aids should generate the necessary format automatically.

Comment: It is not sufficient merely to apply computer checking to validate formats generated by the user.

See also: 5.0-2, 5.1-2.

-4 • Automatic Message Routing -4

When data are transmitted to a standard group of recipients, computer aids should generate the distribution lists and necessary address headers.

Comment: Users might sometimes wish to modify standard distributions, or the distribution for any particular message. Appropriate review/change procedures should be provided.

See also: 5.0-3.

-5 • Automatic Message Initiation -5

When standard messages must be transmitted following data change, as when a computer is monitoring external events, computer aids should initiate transmission automatically.

Example: Many operations-monitoring tasks might benefit from automatic generation of messages to report routine events.

Comment: Automatic transmission of routine messages will reduce the workload on the user, and help ensure timely reporting. However, users should be able to monitor automatic message initiation, and may sometimes wish to modify the logic for automatic message initiation. Appropriate review/change procedures should be provided.

Users may need to control the sequence, content, format, routing, timing, etc., of data transmissions.

-1 • Functional Wording -1

Choose functional wording for the terms used in controlling data transmission, for data specification, message routing and initiation, etc., so that those terms will match the user's job-oriented terminology.

Example: A user should be able to address messages to other people or agencies by name, without concern for computer addresses, communication network structure and routing.

Comment: In general, a user should not have to learn the technical details of communication protocols, codes for computer "handshaking", data format conversion, etc., but should be able to rely on the computer to handle those aspects of data transmission automatically.

See also: 3.1.5-3, 3.1.5-5, 5.0-3.

-2 • Flexible Data Specification -2

Provide users with flexible means for specifying data to be transmitted.

Comment: When sending, a user may wish to specify data by display name or file name, either all or a designated part, or by defined data category. When receiving, a user may wish to accept data from specified sources, and/or by defined data categories.

See also: 5.0-6.

-3 o Receipt by Priority -3

In applications where incoming messages will have different degrees of urgency, i.e., different implications for action by their recipients, permit users to specify destinations (devices) for receiving messages of different priority.

See also: 5.2-4, 5.6-6.

Users may need to specify data source and (device) destination when receiving messages.

-1 • Source Selection -1

For receiving data, permit users to specify from what sources data are needed, and/or will be accepted.

Comment: Computer-mediated message handling offers the potential for screening out the electronic equivalent of "junk mail" or "crank calls". A user might be selective in specifying the people or organizations from which messages will be received.

Comment: Source specification may be in terms of data files, or other users, or external sources. Standard sources may be specified as a matter of routine procedure, with special sources designated as needed for particular transactions.

-2 • Destination Selection -2

When receiving data, permit users to choose the medium of message receipt, i.e., which type of device (files, display, printer) will be the local destination.

Comment: Data might be received directly into computer files, or might be routed to an electronic display for quick review, or to a local printer for hard-copy reference purposes.

Comment: Device destination might be specified differently for different types of messages, or for messages received from different sources.

Comment: When transmitted data are received via display, care should be taken to queue incoming messages, so that they will not interfere with other data processing.

See also: 5.0-6, 5.6-5.

-4 • Assignment of Priority -4

When messages will have different degrees of urgency, i.e., different implications for action by their recipients, then permit the sender of a message to designate its relative priority, or else have the computer assign priority automatically.

See also: 5.3-3, 5.6-6.

-5 • Message Printing -5

Permit users to transmit displayed data to a local printer in order to make hard-copy records.

Exception: In some applications, security constraints make printed records inadvisable.

Comment: User requirements for printed data are often unpredictable to system designers, and so a general printing capability should be provided.

Comment: Printing may be regarded as a special case of data transmission, where no other users are necessarily involved.

Reference: EG 4.2.14; MS 5.15.9.2.

See also: 2.5-4, 2.5-5, 6.2-7, 6.4-5.

Users may need to specify data source and destination when sending messages.

-1 • Source Selection -1

For sending data, permit users to choose whether to transmit from data displays, or directly from computer-stored data files.

Comment: User-initiated transmission from displays will permit the user to review and annotate messages before sending them.

-2 • Destination Selection -2

For sending data, permit users to specify the destination(s) to which data will be sent.

Exception: In bus communication systems, it might be desirable to permit content-driven communication, where potential recipients can request all messages on particular topics, whether or not those messages are specifically addressed to them.

Comment: Specification of destination may be in terms of data files, or other users, or external destinations, including remote printers. Standard destinations may be specified as a matter of routine procedure, with special destinations designated as needed for particular transactions.

See also: 5.0-6.

-3 • Status Information -3

For sending data, provide users access to status information concerning the identity of other system users currently on-line, and the availability of communication with external systems.

Comment: Such status information may influence the user's choice of destinations for data transmission.

See also: 4.1-6, 4.1-8.

SECTION 6

DATA PROTECTION

Data protection attempts to ensure the security of computer-processed data from unauthorized access, from destructive user actions, and from computer failure. With increasing use of computer-based information systems, there has been increasing concern for the protection of computer-processed data. Data protection is closely allied with other functional areas. The design of data entry, data display, sequence control, user guidance, and data transmission functions can potentially affect the security of the data being processed. In many applications, however, questions of data protection require explicit consideration in their own right.

Data protection must deal with two general problems. First, data must be protected from unauthorized access and tampering. This is the problem of data security. Second, data must be protected from errors by authorized system users, in effect to protect users from their own mistakes. This is the problem of error prevention.

Design techniques to achieve data security and to prevent user errors are necessarily different, but for both purposes the designer must resolve a fundamental dilemma. How can the user interface be designed to make correct, legitimate transactions easy to accomplish, while making mistaken or unauthorized transactions difficult? In each system application, a balance must be struck between these fundamentally conflicting design objectives.

Concern for data security will take different forms in different system applications. Individual users may be concerned with personal privacy, and wish to limit access to private data files. Corporate organizations may seek to protect data related to proprietary interests. Military agencies may be responsible for safeguarding data critical to national security.

The mechanisms for achieving security will vary accordingly. Special passwords might be required to access private files. Special log-on procedures might be required to assure positive identification of authorized users, with records kept of file access and data changes. Special confirmation codes might be required to validate critical commands.

At the extreme, measures instituted to protect data security may be so stringent that they handicap normal system operations. Imagine a system in which security measures are designed so that

every command must be accompanied by a continuously changing validation code which a user has to remember. Imagine further that when the user makes a code error, which can easily happen under stress, the command sequence is interrupted to re-initiate a user identification procedure. In such a system, there seems little doubt that security measures could reduce operational effectiveness.

In recent years, computer security measures have concentrated increasingly on automatic means for data protection, implemented by physical protection of computing equipment and by tamper-proof software. Automation of security makes good sense. If data security can be assured by such means, there will be less need to rely on fallible human procedures. And, of course, user interface design will be that much easier.

It seems probable, however, that absolute data security can never be attained in any operational information system. There will always be some reliance on human judgment, as for example in the review and release of data transmissions, which will leave systems in some degree vulnerable to human error. Thus a continuing concern in user interface design must be to reduce the likelihood of errors, and to mitigate the consequences of those errors that do occur.

Like data security, error prevention is a relative matter. An interface designer cannot reasonably expect to prevent all errors, but frequent user errors may indicate a need for design improvement. Data protection functions must be designed 1) to minimize the entry of wrong data into a system; 2) to minimize mistakes that make wrong changes to stored data; and 3) to minimize the loss of stored data. In considering these objectives, prevention of catastrophic data loss is vital for effective system operation, but all three aspects of data protection are important.

Data entry and change transactions are, of course, frequently performed by system users. Careful interface design can help prevent many errors in those transactions, by providing automatic data validation and reversible control actions, as described in previous sections of these design guidelines. But the designer needs a good deal of ingenuity in applying guidelines within the context of each data handling job.

The use of job context for computer validation of user inputs is best illustrated by example. As one such example, consider the following discussion of data entry error prevention that was published in a local newspaper, which suggests ways to reduce billing errors:

> . . . designers of applications systems have resorted to a number of strategies to minimize ill effects and keep the errors from escaping into the world at large. The commonest of these is the process known as 'verification,' which in its simplest form, means instructing the system to respond to input with the figurative question, 'Do you really mean that?'
>
> That is, when a data[-entry] operator enters an amount, or name, or serial number -- the system draws attention to the just-typed item (by causing it to flash on-screen, for example). The operator then is supposed to take a good hard look at the item and press a verification key if the data is correct.
>
> Better yet, what you need is for the system to do some checking on its own . . . to a certain extent, it can use internal evidence (and the percentages) to perform its own verification.
>
> Here's a simple strategy that, though currently used to some extent, will some day become universal, one hopes. It's good because it relies on an understanding of human habits.
>
> Let's take billing again. Most times, when you pay a bill, you pay either the minimum amount due or the full balance. Suppose we instruct the machine to compare the operators entry of 'amount paid' with the minimum due and with the full balance for that particular account. If the entry is equal to one or the other, pass it on through. It's very likely correct. If there's a discrepancy, discontinue entry and signal the operator to check the amount.
>
> And it does so optimally: the right ones pass through with minimum slow down, the potential wrong ones get the attention.
>
> (Bertoni, 1982)

Similar reasoning might be applied in other data entry jobs. Once data are correctly entered into a computer, the emphasis shifts to prevention of unwanted changes to the data, including the extreme form of change represented by data loss. Stored data must be protected from unreliable computer operation and also from system users. Advances in computer technology, with less volatile memory, automatic archiving to back-up data stores, and redundant processing facilities, have significantly reduced the hazard of data loss

resulting from machine failure. What remains is to reduce the hazard of human failure.

In the interface design guidelines presented here, the primary concern is for the general users of computer systems. But data protection from human error requires consideration in other aspects of system design and operation. In particular, the expert operators who maintain and run the computer system must assume a large responsibility for data protection.

Consider the following example. In one computer center, an operator must enter a command "$U" to update an archive tape by writing a new file at the end of the current record, while the command "$O" will overwrite the new file at the beginning of the tape so that all previous records are lost. A difference of one keystroke could obliterate the records of years of previous work. Has that ever happened? Yes, it has. If an error can happen, then it probably will happen.

In that respect, expert computer operators are just like the rest of us. When tired, hurried, or distracted, they can make mistakes. And not all computer operators are experts. Some are still learning their jobs, and so may be even more error-prone. Careful design and supervision of operating procedures is needed to minimize data processing errors.

If data loss from machine failure and data loss from faulty system operation are minimized through careful design, then the most serious threat to data protection is the system user. This is especially true in applications where the user must participate directly in establishing and maintaining stored data files. Means must be found to protect files from inadvertent erasure.

Some difficult design trade-offs may be required. As an example, consider a possible design guideline that might say that a user should not be allowed to change or delete data without first displaying the data. In a file deletion transaction it would usually be impractical to force a user to review the entire file. One might imagine displaying just the first page of a file nominated for deletion, and requiring the user to CONFIRM the DELETE action. But even that would be disruptive in many circumstances.

As a fall-back position, we might recommend that when a file has been nominated for deletion enough descriptive information should be displayed about that file so that a reasonably attentive user can determine whether that file should be deleted. The issue is how to ensure that the user knows what he/she is doing.

When a user selects a file for deletion, at least as much information should be provided as when a user selects a file for display and editing. Thus, if an on-line index of displayable files contains a line of information about each one, perhaps including name, description, size, and currency (date last changed), then such information would also be appropriate in prompting the CONFIRM action for file deletion:

CONFIRM DELETION OF THE FOLLOWING FILE:

USIplan 5-year plan for USI effort 3 pages 11-25

Any required confirmation procedure, of course, will tend to slow file deletion, in accord with the general guideline that destructive actions should be made difficult. Where is the trade-off when destructive actions are also frequent? What about the user who wishes to scan a file index and delete a series of files? Must each separate DELETE action be confirmed? Unless DELETE actions are easily reversible, the answer for most users is that an explicit confirmation probably should be required for each file deletion. When a user must undertake a series of file deletions, the repetitive nature of the task may increase the risk of inadvertent deletion, and so increase the need for CONFIRM protection.

Explicit DELETE commands are not the only actions that can result in accidental file erasure. In some systems, it is possible to overwrite a stored file with whatever data are currently on-line, in "working" storage. Used properly, this capability permits desired editing and replacement of files. A user might call out a file, make changes to it, and then store it again under its own name.

Used improperly, this capability risks file deletion. A user might call out and edit a file, but then absent-mindedly store it with the name of another file, thus overwriting whatever data had been previously stored in that other file. Such a hazard requires just as much protection as an outright DELETE action, or perhaps even more since the danger is more subtle. In effect, an explicit CONFIRM action should be required whenever a user attempts to store a data file under the name of any other file already stored in the system. The prompt for confirmation might read something like this:

CONFIRM OVERWRITING THE CURRENT FILE OF THIS NAME:

SCG sequence control guidelines 45 pages 10-08

It is interesting that many systems do not require this kind of selective confirmation. One well-known system requires user

confirmation of every overwrite action, even in the common case where an edited file is being stored by the same name to replace its previous version. Thus the CONFIRM action itself becomes routine, and no longer provides any significant protection to a forgetful user. Another system avoids the problem by the rigid expedient of allowing a user to store an edited file only under its last previous name, which is safe but sometimes inconvenient.

To some extent a wary user can protect himself/herself by careful selection of file names, trying to ensure that any file name is descriptive of file content, and also distinctive in comparison with the names of other files. In practice, that goal is hard to achieve. Users often work with groups of files dealing with different aspects of a common topic. For such files, if names are descriptive they will tend to be similar rather than distinctive. If file names are made longer in order to become more distinctive, then those longer names may reduce efficiency when using file names for storage and retrieval.

In systems where there is no effective on-line protection against inadvertent file deletion or replacement, either a user must be exceedingly careful, or else the system must provide effective off-line procedures to recover from archive records an earlier version of an accidentally erased file. Neither alternative is entirely satisfactory. Even a careful user will make mistakes. And archives will not protect a user from loss of current work.

A better solution can be provided by on-line computer aids to make user actions reversible. In effect, user interface software should be designed to permit users who notice unintended deletions to retrieve lost files by taking an UNDO action. Some current systems provide such an UNDO capability, permitting users to change their minds and to correct their more serious mistakes.

There must be, of course, some practical time limit to the reversibility of data processing. A user might be able to UNDO the last previous deletion, or perhaps even all deletions made during the current working session, but there seems no feasible way to make it easy to undo a particular deletion made days ago and now regretted. Moreover, reversibility will not help a user who does not notice that a mistake has been made. So even where an UNDO capability is available, other aspects of the user interface must still be carefully designed.

The guidelines proposed in the following pages illustrate the range of topics to be considered in this area, and the general need for many kinds of data protection. These guidelines draw heavily from recommendations already made in previous sections, as indicated

by extensive cross referencing. In this new context of data protection, some previous guidelines have been slightly reworded, others preserved intact. They are repeated here for the convenience of a designer who must review all material pertinent to data protection functions.

These guidelines do not resolve the fundamental design dilemma discussed above, namely, how to make a system easy to use but hard to misuse. The guidelines do, however, indicate where design decisions must be made.

- Data protection refers to ensuring data security from unauthorized access, and also from destructive user actions.
- Procedures for user indentification should be as simple as possible consistent with data protection.
- Procedures to prevent data access by unauthorized users should not impede authorized access to those data.
- Procedures established to prevent data entry/change by unauthorized users should not impede authorized data use.
- Procedures for sending and receiving messages should provide protection for the data transmission process.
- Measures to prevent data loss should include protection from user errors as well as from computer failure.
- Changes to the software design of data protection functions may be needed to meet changing operational requirements.

Objectives:

Effective data security
Minimal entry of wrong data
Minimal loss of needed data
Minimal interference with data handling tasks

Data protection refers to ensuring data security from unauthorized access, and also from destructive user errors.

-1 • Automatic Security Measures -1

Whenever possible employ automatic measures to protect data security, relying on computer capabilities rather than on fallible human procedures.

-2 • Consistent Procedures -2

User interface design should provide consistent procedures for data transactions, including data entry and error correction, data change and deletion.

Comment: Consistent procedures will reduce the likelihood of user confusion and error, and are especially important for any transaction that risks data loss.

Reference: BB 1.2.1, 2.1.5.

See also: 4.0-1, 5.0-1, 6.5-6.

-3 • Control by Explicit User Action -3

The computer should change data only as a result of explicit actions by a user, and not initiate changes automatically.

Exception: A computer might perform cross-file updating automatically, following data change by a user.

Exception: In an operations monitoring situation, a computer might accept data changes automatically from external sources (sensors), if appropriate software is incorporated to ensure input validation and data protection.

Comment: In effect, a computer should not initiate data changes unless requested (and possibly confirmed) by a user. Interface designers are sometimes tempted to contrive "smart shortcuts" in which one user action may automatically produce several other associated data changes, perhaps saving the user a few keystrokes. Since such shortcuts cannot generally be made standard procedures, they will tend to confuse novice users, and so may pose a potential threat to data protection.

See also: 1.0-8, 1.1-4, 3.0-5, 3.1.3-6, 3.5-6, 4.0-2, 5.0-7.

-4 • Feedback for Mode Selection -4

When the result of user actions will be contingent upon prior selection among differently defined operational modes, provide a continuous indication of the current mode, particularly when user inputs in that mode might result in unintended data loss.

Example: If a DELETE mode is used to edit displayed data, some indication of that mode should be displayed to users.

Comment: A user cannot be relied upon to remember prior actions. Any action whose results are contingent upon previous actions represents a potential threat to data protection.

Reference: BB 4.3.4; MS 5.15.5.5.

See also: 4.2-8, 6.5-13.

-5 • Appropriate Response to All Entries -5

User interface design should deal appropriately with all possible control entries, correct and incorrect, without introducing unwanted data change.

Comment: The interface designer must try to anticipate every possible user action, including random keying and perhaps even malicious experimentation. The user interface must be "bullet-proofed" so that an unacceptable entry at any point will produce no more significant computer response than an error message.

Reference: BB 5.1; MS 5.15.2.1.2; PR 4.12.4.5.

See also: 3.5-1.

-6 • User Review and Editing of Entries -6

For both data entry and control entry, allow users to edit composed material before initial entry, and also before any required re-entry.

Comment: This capability will allow a user to correct many entry errors before computer processing. When errors are made, the user will be able to fix them without having to regenerate correct items and risk introducing further errors.

Reference: BB 5.2.1; EG 5.4; Neal and Emmons, 1982.

See also: 1.4-2, 3.5-2, 4.3-14.

-7 • Resolving Ambiguous Entries -7

The computer should question users as necessary to resolve data entries or control entries that require interpretation, and should aid users in correcting erroneous inputs.

Example: The computer might help resolve an ambiguous abbreviation by asking the user to select among displayed alternatives.

See also: 1.0-22, 6.5-11.

-8 • User Warned of Threats to Security -8

Provide messages and/or alarm signals to warn users of potential threats to data security.

Reference: EG 2.1.3.

See also: 4.3-16, 4.3-17, 6.5-17.

Procedures for user identification should be as simple as possible, consistent with data protection.

-1 • Easy LOG-ON -1

Design the LOG-ON process and procedures for user identification to be as simple as possible consistent with protecting data from unauthorized use.

Comment: Authentication of user identity is generally not enhanced by requiring a user to enter routine data such as terminal, telephone, office or project numbers. In most organizations, those data can readily be obtained by other people. If verification of those data is needed, the user should be asked to review and confirm currently stored values in a supplementary procedure following LOG-ON.

Reference: MS 5.15.7.5.f.

See also: 1.8-6.

-2 o Prompting LOG-ON -2

Design the LOG-ON process to provide prompts for all user entries, including passwords and/or whatever other data are required to confirm user identity and to authorize appropriate data access/change privileges.

Reference: EG 4.2.11.

-3 • User Choice of Passwords -3

When passwords are required, allow users to choose and/or to change their own passwords.

Comment: The password chosen by a user will generally be easier for that individual to remember. User choice is especially helpful when passwords must be periodically changed.

Measures to prevent data loss should include protection from user errors as well as from computer failure.

-1 • Protecting Data from Computer Failure -1

Design information systems to minimize data loss from computer failure.

Example: Depending upon the criticality of the application, different protective measures may be justified, including periodic automatic archiving of data files, maintenance of transaction logs for reconstruction of recent data changes, or even provision of parallel "backup" computing facilities.

Comment: An automatic capability is needed because users cannot be relied upon to remember to take necessary protective measures.

Comment: Though not strictly a feature of user interface design, reliable data handling by the computer will do much to maintain user confidence in the system. Conversely, data loss resulting from computer failure will destroy user confidence, and reduce user acceptance where system use is optional.

Reference: MS 5.15.4.6.3.

-2 • Protecting Data from Other Users -2

Protect all user transactions and data from actions by other users.

Comment: When one user's actions can be interrupted by another user, as in defined emergency situations, that interruption should be temporary and nondestructive. The interrupted user should subsequently be able to resume operation at the point of interruption without data loss.

Reference: MS 5.15.4.6.5.

See also: 3.0-19.

-3 • Saving Sent Data until Receipt is Confirmed -3

When data are being sent, the computer should save a copy of the transmission automatically until correct receipt is confirmed (and possibly longer in many applications).

Comment: The primary objective is to prevent irretrievable data loss during transmission. For many system applications, however, the originator of a message will probably want to retain a copy in any case, and will prefer that any subsequent deletion be handled as a separate transaction, distinct from data transmission.

-4 • Queuing Received Messages -4

When data are being received, the computer should queue incoming messages as necessary to ensure that they will not disrupt or destroy any ongoing data transactions by a user.

Comment: In general, incoming data should not replace existing data until reviewed by the user. Exceptions must be made, however, in applications where automatic updating of current situation data is required for effective user monitoring, as in air traffic control systems. The difference there is that data updating is the primary purpose of the system, and the user knows what is going on.

See also: 5.6-4.

-5 • Printing Messages -5

Insofar as possible within constraints of data security, allow users to generate printed copies of transmitted data, including messages sent and received.

Comment: User requirements for printed data may be unpredictable, and restrictions on data printout may hinder task performance. Rather than restrict printout, appropriate procedures should be established for restricting further distribution of printed messages.

See also: 2.5-9, 5.3-3, 6.2-7.

Procedures for sending and receiving messages should provide protection for the data transmission process.

-1 • Automatic Protection of Transmitted Data -1

Ensure that whatever measures are adopted to protect data during transmission -- e.g., encryption, parity checks, buffering until acknowledgment of receipt, etc. -- will be applied by the computer automatically, without the need for any explicit action by a user.

Comment: Users are fallible, and cannot be relied upon to participate quickly and accurately in the mechanisms of data transmission, whereas this is the sort of thing that computers can do well. A user might be asked to supply an encryption key, but the computer should handle any actual encryption process.

See also: 6.0-1.

-2 • User Review of Data before Transmission -2

When human judgment may be required to determine whether to release data for transmission, provide users (and/or an authorized supervisor) some convenient means to review outgoing messages and confirm their release before transmission.

Comment: Sometimes message release may require coordination among several reviewers in the interests of data protection.

See also: 5.4-6.

-19 • *Displaying Data to be Changed* -19

When a user requests change (or deletion) of a stored data item that is not currently being displayed, then the computer should offer to display both the old and new values so that the user can confirm or nullify the change before the transaction is completed.

Comment: This practice will tend to prevent inadvertent change, and may be particularly useful in protecting delete actions.

See also: 1.0-13, 6.5-10.

-20 • User Confirmation of Destructive Actions -20

Require users to take explicit action to confirm doubtful and/or potentially destructive data change actions before they are accepted by the computer for execution.

Comment: A requirement to take an explicit CONFIRM action will direct user attention to questionable data changes, and help the user avoid the consequences of thoughtless errors.

Reference: BB 5.6; MS 5.15.7.5.b.

See also: 1.3-33, 1.3-35, 4.3-15, 4.3-16, 6.0-8, 6.5-18.

-16 o Automatic Data Generation -16

When routine/redundant data are available in the computer, access or derive those data automatically for user review, rather than requiring entry by the user.

Comment: This represents an exception, in the interests of improved data accuracy, to the general recommendation that data entry/change should occur only as a result of explicit user actions. Automatic data generation by the computer, where it can be based on derived values or cross-file updating, will be faster and more accurate, with risk of error being introduced in re-entry by the user. In effect, having a computer do automatically what the user may do poorly is here regarded as a form of data protection.

Reference: BB 2.4.2; MS 5.15.2.1.6.

See also: 1.8-6, 1.8-7, 1.8-9, 1.8-10.

-17 • Validation of Changed Data -17

Provide data validation software which will check for erroneous or doubtful values for data changes as well as data entries.

Comment: Do not rely on the user always to make correct entries. When validity of data entries can be checked automatically, such computer aids will help improve accuracy of data entry.

Reference: MS 5.15.2.1.5, 5.15.7.3; PR 4.12.4.

See also: 1.7-1.

-18 o Cross Validation of Changed Data -18

Whenever possible, provide automatic cross validation of data entries and changes to ensure that each data item is logically consistent with other related data items.

Comment: Such cross checking is one of the potential advantages of on-line data handling, helping users detect logical errors.

Reference: MS 5.15.7.3; PR 4.12.5.

See also: 1.4-1, 1.7-1, 6.3-9.

-13 • Flexible BACKUP for Error Correction -13

Allow users to return easily to previous steps in a transaction sequence in order to correct an error or make any other desired change.

Reference: MS 5.15.7.7.

See also: 3.5-13.

-14 • Explicit Entry of Corrections -14

Require users to take an explicit ENTER action to request computer processing of corrections to wrong data; this should be the same action that was used to enter the data originally.

Reference: MS 5.15.7.9; PR 4.12.6.

See also: 3.5-6, 6.0-3.

-15 • Data Verification by User Review -15

When verification of prior data entries is required, allow users to review and confirm the data, rather than requiring re-entry of the data.

Comment: For routine verification, data review by the user will be quicker than re-entry, with less risk of introducing new errors.

Comment: For special verification, as when computer processing has detected doubtful and/or discrepant data entries, the user should be alerted with an appropriate advisory message.

See also: 1.8-8, and Section 4.3.

-10 • User Editing of Data Before Entry -10

Allow users to correct keyed data items (or commands) prior to taking an explicit ENTER action.

Comment: Easy correction prior to entry will avoid the need for computer processing of user-detected errors.

Comment: Error correction procedures should permit back-spacing with a nondestructive cursor to the point of error, re-keying the corrected item, followed by an ENTER action without any further cursor positioning.

Reference: EG 5.4; Neal and Emmons, 1982.

See also: 1.4-2, 3.5-2, 6.0-6.

-11 o Immediate Error Correction -11

When an error in data entry/change is detected by the computer, allow the user to make an immediate correction.

Comment: Immediate corrections will be made more easily and accurately when the data (e.g., source documents) are still available to the user.

Reference: EG 5.7; MS 5.15.7.7.

See also: 1.7-6, 3.5-12.

-12 o User Editing of Entry Errors -12

Following error detection, require users to re-enter only that portion of a data entry that was not correct.

Comment: If the user must re-enter an entire data set to correct one wrong item, then there will be the risk of new errors in previously correct items.

Reference: BB 5.2.1; EG 4.2.3, 5.4; MS 5.15.7.1.

See also: 4.3-14, 6.0-6.

-8 • Explicit User Actions -8

Data entry/change should result only from an explicit ENTER action by the user, and should not be a possibly unrecognized side effect of other actions.

Example: A dual-activation lightpen that permits pointing at an item without data entry/change unless some further explicit action is taken.

Exception: Automatic generation of routine, redundant or derived data.

Comment: Explicit actions will help direct user attention to data entry/change, and reduce the likelihood of thoughtless errors.

Reference: MS 5.15.2.1.4.

See also: 1.0-8, 1.1-4, 3.0-5, 3.1.3-6, 4.0-2, 6.0-3.

-9 o Single Entry of Related Data -9

Allow users to enter logically related items, as in a form-filling dialogue, with a single, explicit action at the end of the sequence, rather than entering each item separately.

Comment: This practice permits user review and possible data correction prior to entry. It will also permit efficient cross validation of related data items by the computer.

See also: 1.4-1, 6.3-18.

-4 • Segregating Real from Simulated Data -4

When simulated data are processed and stored, as for on-line user training, ensure that changes to simulated data are processed separately and do not affect real data.

Reference: BB 6.4.

See also: 4.4-25, 6.2-2, 6.5-4.

-5 • Emphasizing Accuracy -5

When accuracy of data entry/change is more important than speed, display formats and user guidance should emphasize the accuracy requirement.

Comment: Slow but correct initial entry of data will take less time than a hasty entry that must later be corrected.

Reference: EG 7.2.

-6 • Simple Procedures -6

Make procedures for data entry/change as simple as possible, by following general guidelines for data entry.

Example: Allow users to enter short rather than long items, do not require users to enter leading zeros or count blanks, etc.

Comment: Simple procedures will help ensure accuracy in data entry/change transactions.

See also: 1.0-14, 1.0-26, 1.0-27, 3.1.5-15.

-7 o Displaying Default Values -7

Display all currently operative data entry default values for user confirmation prior to processing by the computer.

Reference: BB 2.1.10.

See also: 1.8-3, 1.8-4.

Procedures estabished to prevent data entry/change by unauthorized users should not impede authorized data use.

-1 • Single Authorization for Data Entry/Change -1

Establish user authorization for data entry/change at initial LOG-ON; do not require further authorization when a user attempts particular data entry/change transactions.

See also: 6.1-5.

-2 • Data Entry/Change Transaction Records -2

When necessary for data protection, allow users (and/or a system administrator) to request a record of data entry/change transactions.

Comment: Transaction records might, of course, be maintained for purposes of user guidance as well as for data protection, as recommended elsewhere.

See also: 3.4-3, 4.4-18, 4.5-3.

-3 • Protection from Data Change -3

When data must not be changed, maintain computer control over the data and do not permit users to change controlled items.

Comment: Never assume compliance with instructions by the user, who may attempt unwanted changes by mistake, or for curiosity, or to subvert the system.

Comment: Similar considerations also dictate routine protection of display formatting features, as recommended elsewhere.

Reference: MS 5.15.4.3.12.

See also: 1.1-23, 1.4-7, 2.0-9, 6.2-5.

-6 • Easy Display Suppression -6

When confidential information is displayed at a work station that might be viewed by casual onlookers, provide the user with a rapid, easy means of temporarily suppressing a current display if its privacy is threatened, and then resuming work later.

Comment: Such a capability is sometimes called a "security pause". If quick display suppression is required, a function key should probably be provided for that purpose. To retrieve a suppressed display and resume work, the user should probably be required to make a code entry such as a password, in the interests of data protection.

Comment: Actually, a suppressed display should not be entirely blank, but should contain an appropriate message indicating current status, e.g., DISPLAY IS TEMPORARILY SUPPRESSED; ENTER PASSWORD TO RESUME WORK.

See also: 3.3-8, 4.2-1, 6.1-5.

-7 • Printing Protected Data -7

Within constraints of data security, allow users to generate printed copies of displayed data.

Comment: User requirements for printed data are often unpredictable, and printing restrictions may handicap task performance. Rather than restrict printing, establish appropriate procedures for restricting further distribution of data printouts.

Reference: BB 4.4.6; EG 4.2.14; MS 5.15.9.2.

See also: 2.5-9, 5.2-5, 6.4-5.

-8 • Automatic Records of Data Access -8

When records of data access are necessary, the computer should keep those records automatically; do not require users to take record keeping actions.

Comment: Even cooperative, well-intentioned users can forget to keep manual logs of data access, and will resent the time and effort required to keep such logs. Subversive users, of course, cannot be expected to provide accurate records.

See also: 4.5-4.

-3 • User Editing of Displayed Data -3

If users are authorized to access data for display, allow those users also to make changes to displayed data; where data changes are not authorized, indicate that on the display.

Comment: If users can usually make additions and/or corrections to displayed data, then any exception to that practice may confuse a user.

See also: 2.0-8.

-4 • Protecting Displayed Data -4

When protection of displayed data is essential, maintain computer control over the display and do not permit the user to change controlled items.

Comment: Never assume compliance with instructions by the user, who may attempt unwanted changes by mistake, or for curiosity, or to subvert the system.

Reference: EG 3.4.8; MS 5.15.4.3.12.

See also: 2.0-9.

-5 o Protecting Display Formats -5

Protect display formatting features, such as field labels and delimiters, from accidental change by users.

Reference: BB 1.8.13; MS 5.15.3.1.1.c.

See also: 1.1-23, 1.4-7.

Procedures established to prevent data access by unauthorized users should not impede authorized access to those data.

-1 • Single Authorization for Data Access -1

Establish user authorization for data access at initial LOG-ON; do not require further authentication when a user requests display of particular data.

See also: 6.1-5.

-2 • Displayed Security Classification -2

When displayed data are classified for security purposes, include a prominent indication of security classification in each display.

Comment: This practice will serve to remind users of the need to protect classified data, both in access to the display itself and in any further dissemination of displayed data.

Comment: In applications where either real or simulated data can be displayed, a clear indication of simulated data should be included as part of the classification label.

Comment: Where a display includes several partitioned "windows" of data from different sources, it may be necessary to label security classification separately for each window. Under those conditions, some form of auxiliary coding (e.g., color coding) may help the user distinguish among different classes of data.

See also: 6.5-4.

-4 o Private Entry of Passwords -4

When a private password must be entered by a user, that entry should be covert, i.e., should not be displayed.

Comment: This represents an exception to the general recommendation that all entries should be displayed.

Reference: MS 5.15.2.2.3.

See also: 1.0-2.

-5 • Continuous Recognition of User Identity -5

Once a user's identity has been authenticated, whatever data access/change privileges are authorized for that user should continue throughout a work session.

Exception: In special instances a user's data access/change privileges might reasonably change as a result of succeeding transactions, e.g., if computer analysis indicated suspicious or otherwise abnormal behavior.

Exception: A user might reasonably be required to repeat procedures for authentication of identity following some specified period of inactivity at the work station, or when resuming work after a requested suspension of activity.

Comment: If a previously identified user later is required to take separate actions to authenticate data handling transactions, such as access to particular files or issuance of particular commands, the efficiency of system operations may be degraded. Where continuous verification of user identity seems required for data protection, perhaps some automatic means of identification can be devised for that purpose.

Reference: Symons and Schweitzer, 1984.

See also: 3.3-10, 6.2-1, 6.2-6, 6.3-1.

-3 • Protecting Data from Interrupt Actions -3

When a user action interrupts data processing -- e.g., BACKUP, CANCEL, RESTART, ABORT, etc. -- there should be no change or loss of stored data except for those changes specified by the action taken.

Comment: If an interrupt action will cause extensive data change -- e.g., ABORT, UNDO -- require the user to confirm the action before processing.

Reference: BB 4.7.

See also: 3.3-6, 6.5-17 thru -20.

-4 • Segregating Real from Simulated Data -4

When simulated data and system functions are provided for on-line user training, protect real data and distinguish simulated from actual system operation.

Reference: BB 6.4.

See also: 4.4-25, 6.2-2, 6.3-4.

-5 • Appropriate Ease/Difficulty of User Actions -5

The ease of control actions by the user should match desired ends; make frequent or urgent actions easy to take, but make potentially destructive actions sufficiently difficult that they require extra user attention.

-6 • Standard Procedures -6

Provide standard procedur s for accomplishing different types of transactions, to facilitate user learning and efficient system operation.

Comment: Standard procedures will help the user build consistent operational habits, reduce confusion, and reduce the likelihood of user errors.

Reference: BB 1.2.1.

See also: 4.0-1, 6.0-2.

-7 • Disabling Unneeded Controls -7

The computer should temporarily disable function keys (and other control devices) when they are not needed for current control entry, especially when they may have destructive effects.

See also: 3.1.4-10, 3.2-10.

-8 o Protecting Controls -8

If activation of function keys (and other control devices) may result in data loss, locate and/or physically protect them to reduce the likelihood of accidental activation.

Reference: MS 5.15.4.1.2.

See also: 3.1.4-16.

-9 • Data Entry Independent of Cursor Placement -9

An ENTER action for multiple data items should result in entry of all items, regardless of where the cursor is placed on the display.

Comment: A user may choose to move the cursor back in order to correct earlier data items, and cannot be relied upon to move the cursor forward again. The computer should ignore cursor placement in such cases.

See also: 1.1-24.

-10 • Displaying Data to be Changed -10

When a user requests change (or deletion) of a stored data item that is not currently being displayed, then the computer should offer to display both the old and new values so that the user can confirm or nullify the change before the transaction is completed.

Comment: This practice will tend to prevent inadvertent change, including changes resulting in loss of needed data. User attempts at selective data change without displayed feedback will be error prone.

Comment: For delete actions involving significant amounts of data, such as entire files, display of all the data will probably not be feasible. In such instances, the user should be clearly warned of potential data loss and required to confirm that destructive action.

See also: 1.0-13, 6.3-19, 6.5-17, 6.5-18.

-11 • User Review of Interpreted Commands -11

When an ambiguous command entry is interpreted by the computer, and especially if that interpreted command threatens data loss, allow the user to review and confirm a displayed interpretation of the command before it is executed.

Comment: In many systems, of course, the computer will execute a recognized command or else simply display an error message if a command is not recognized. The concern here is for a possible intermediate case, where the computer is programmed to try to interpret an "almost recognized" input by modifying it to become a proper command. Although that practice may aid users in some instances, a computer interpretation will not always match the user's intent.

See also: 3.1.5-19, 6.0-7, 6.5-18.

-12 • Protective Defaults -12

If automatic defaults are provided for control entries, ensure that those defaults will protect against data loss, or at least not contribute to the risk of data loss.

Example: When requesting a printout of filed data, one option may be to delete that file after printing; the default value for that option should automatically be set to NO whenever printing options are presented to a user for selection.

-13 • Explicit User Action to Select Destructive Modes -13

Require users to take explicit action to select any operational mode that might result in data loss; the computer should not establish destructive modes automatically.

Example: In text editing, if a user takes a DELETE action, that in itself should not establish a continuing DELETE mode.

Comment: In many applications, it may be better not to provide any destructive mode. Instead of providing a DELETE mode, for example, require that DELETE be a discrete action subject to confirmation by the user. User interface design must determine the proper balance here between data protection and operational efficiency.

See also: 1.3-27, 4.2-8, 6.0-4.

-14 • Protecting Data from User Error -14

Require users to take at least two separate actions to implement any destructive command; no single user action should cause significant change or loss of stored data, such as deleting an entire data file.

Example: Enter next command: D

If deleted, all data in this file will be lost.
Enter YES to confirm deletion: _____

Comment: Double keying is not sufficient protection. The two user actions must be separated by some computer response.

Reference: BB 5.6; EG 4.2.8; MS 5.15.7.4, 5.15.7.5.b.

See also: 3.5-7, 4.3-16, 6.0-8, 6.3-20, 6.5-18.

-15 • Distinctive File Names -15

When data files may be deleted by name, ensure that the names of different files are distinctive.

Comment: If this guideline is not followed, it is easy for the user to make an error in file storage, by specifying an unintended overwriting of one file with data from a similarly named other file.

Comment: When two or more files are similarly named, the distinctive feature should be near the beginning of the names rather than at the end; in particular, no file name should simply be a truncated version of another.

Comment: In many applications, file naming is a user option, and distinctive naming can be achieved only as a matter of good operational procedure. In such circumstances, perhaps the only feasible aid in the programmed user interface might be a computer-generated advisory message when a proposed new file name is similar (e.g., identical in the first 5 letters) to the name of an existing file.

-16 • Preventing Data Loss at LOG-OFF -16

When a user requests LOG-OFF, the computer should check pending transactions and, if data loss seems probable, should display an appropriate advisory message.

Example: CURRENT DATA ENTRIES HAVE NOT BEEN FILED;
SAVE IF NEEDED BEFORE CONFIRMING LOG-OFF.

Comment: The user may sometimes suppose that a job is done before taking necessary implementing action.

Reference: BB 4.8; MS 5.15.7.5.e.

See also: 3.5-11, 4.3-14.

-17 • Warning Users of Potential Data Loss -17

For conditions requiring (or implying the need for) special user attention to protect against data loss, provide an explicit alarm and/or warning message to prompt appropriate user action.

See also: 3.5-8, 4.3-14, 4.3-17, 6.0-8, 6.5-14.

-18 • User Confirmation of Destructive Actions -18

As well as displaying a warning message, require users to take an explicit further action to confirm a potentially destructive action before it is accepted by the computer for execution.

Reference: BB 5.6; MS 5.15.7.5.b; Foley and Wallace, 1974.

See also: 1.3-33, 1.3-35, 3.1.5-21, 3.5-7, 4.3-16, 6.3-20, 6.5-14.

-19 o Distinctive CONFIRM Action -19

Provide a distinctively labeled CONFIRM function key for user confirmation of doubtful actions.

See also: 3.5-9.

-20 • Reversible Control Actions: UNDO -20

The computer should maintain a record of data changes resulting from current transactions, as necessary to permit users to UNDO immediately preceding control actions that may have caused an unintended data loss.

Comment: UNDO itself should be reversible, so that a second UNDO action will do again whatever was just undone.

Comment: Some version of such an UNDO capability is now often provided in interface design. UNDO represents one more level of data protection, when warning messages and confirmation procedures fail to prevent error, but can only help the user who notices that an error has been made.

Reference: MS 5.15.7.7; Lee and Lochovsky, 1983; Nickerson and Pew, 1971; Shneiderman, 1982.

See also: 1.3-34, 3.5-10.

Changes to the software design of data protection functions may be needed to meet changing operatioal requirements.

-1 • Flexible Design for Data Protection -1

When data protection requirements may change, which is often the case, provide some means for users (or a system administrator) to make necessary changes to data protection functions.

Comment: Data protection functions that may need to be changed include those represented in these guidelines, namely, changes in protective measures regulating user identification, data access, data entry/change, data transmission, and methods of loss prevention.

-2 • Protection from Design Change -2

User interface design should be protected from changes that might impair functions supporting data entry, data display, sequence control, user guidance, data transmission and data protection.

Comment: A trade-off is required between design flexibility, to permit needed improvements to the user interface, and design control, to protect current functions from undesirable changes. Some form of continuing configuration management should be instituted to evaluate changes to user interface design, just as for any other critical system interface.

REFERENCES

Anyone involved in compilation of design guidelines must begin and end by acknowledging the significant contributions of other people. No one person, no matter how wise, can know everything about the complexities of user interface design. Nor will any one person have the perfect judgment, and find the perfect words, to express that knowledge to an interface designer. Thus when we propose guidelines we must build upon the work of others.

This is a good thing. In the introduction to this report, it is argued that all design guidelines are necessarily based in some degree on judgment. If that is so, then guidelines development must properly be a collaborative effort. The collective judgment of many people will often prove sounder than the ideas of just one person.

When many people contribute to guidelines development, we must find ways to acknowledge that contribution. One way is to cite previously published papers that pertain to the guidelines. Citations in this report are represented in the reference list that follows. But in the next several pages we also try to acknowledge more direct contributions to our work.

Many of the user interface design guidelines proposed in this report were not invented here, but derive from the ideas of other people. Where the idea for a guideline came from a particular source, an appropriate reference citation has been included for that guideline. Such citation offers credit where credit is due. More importantly, cited references permit anyone questioning a particular guideline to explore its antecedents, perhaps to gain a better understanding of what is intended.

Citing the source of an idea does not necessarily mean that there is convincing data to support a guideline. Although the references cited here all contain worth-while ideas, only some of these references report results from systematic data collection. Furthermore, citation of references does not necessarily mean that their authors would agree with the wording of guidelines presented here. In some instances, an idea has been borrowed intact. In many more cases, however, ideas have been modified, sometimes drastically, perhaps beyond the intent of their original authors.

In this report, in both text and guidelines, citations of specific references are in conventional form, showing author(s) and publication date. Those references are listed in the pages that follow. The particular format used here for citation and listing of references conforms in most respects to the standard referencing practice recently adopted by the Human Factors Society (1984).

However, four reference sources have been used generally throughout the guidelines. Those sources are cited so frequently that they have been indicated simply by initials:

BB = Brown, Brown, Burkleo, Mangelsdorf, Olsen, and Perkins, 1983

EG = Engel and Granda, 1975

MS = MIL-STD-1472C (as revised), 1983

PR = Pew and Rollins, 1975

These four general references share a common characteristic -- like this report, they are all collections of design guidelines. None of these four general references provide supporting data for their design recommendations, and they need not be consulted for that purpose. The two early reports (EG and PR) have served as a fertile source of ideas for our current guidelines; where those reports are cited here, it means that their early recommendations are still judged to be correct. The two recent reports (BB and MS) have drawn heavily from common sources, including previous editions of the guidelines proposed here; where those reports are cited, it means that their authors have made similar recommendations to those presented here.

The 1975 IBM report by Engel and Granda (EG) was the first widely recognized compilation of user interface design guidelines. That report has provided inspiration and has served as a seminal reference for others working in this field. It is still in demand, and has been reprinted to permit its continued distribution. That report has been cited here 173 times, for some 139 guidelines.

The 1975 BBN report by Pew and Rollins (PR) represents an admirable attempt to propose design guidelines for one particular system application. Its recommendations, however, can readily be generalized for broader application. That report has been cited here 81 times, for 71 guidelines.

The 1983 report by Lin Brown and his colleagues at Lockheed (BB) is a good example of user interface guidelines developed for use as an in-house design standard, but which are available for public reference. That report has been cited here 246 times, for 202 guidelines. Previous editions of the Lockheed report also influenced the formatting of guidelines adopted here, including the use of short titles.

MIL-STD-1472C (MS), in its current revision, is the US military standard for human engineering in system design. This standard has

been cited here 236 times, for 209 guidelines. Of course, guidelines do not carry the same weight as design standards -- guidelines are usually cited for optional application in system development, rather than being imposed contractually. However, there is considerable correspondence between the current military standard and the guidelines proposed here.

Not all ideas for guidelines come from published references. Some of the guidelines proposed here have resulted from discussion with professional colleagues. And the wording of all guidelines has been improved through critical review of earlier published versions. Over the past several years, a number of people have contributed suggestions for improving the guidelines material:

Sara R. Abbott	Union Carbide Corporation
James H. Alexander	Tektronix, Inc.
Christopher J. Arbak	Systems Research Laboratories, Inc.
Arlene F. Aucella	Consultant
J. David Beattie	Ontario Hydro
Leo Beltracchi	US Nuclear Regulatory Commission
C. Marlin Brown	Lockheed Missiles and Space Company
Alphonse Chapanis	Communications Research Laboratory
Hal Cheney	OCLC
Kent B. Davis	Litton Data Command Systems
Robert S. Didner	AT&T
John Dinan	Raytheon Equipment Division
Susan M. Dray	Honeywell, Inc.
Joseph S. Dumas	American Institutes for Research
Sam L. Ehrenreich	AT&T Bell Laboratories
Jeanne Fleming	The MITRE Corporation
Wilbert O. Galitz	Galitz, Inc.
Robert N. Gifford	Northrop Electronics
Susan R. Gilbert	The MITRE Corporation
Nancy C. Goodwin	The MITRE Corporation
Richard M. Kane	Wang Laboratories
Karen L. Kessel	The Koffler Group
Judith R. Kornfeld	Alphatech, Inc.
Jack I. Laveson	Integrated Systems Research
Harold Miller-Jacobs	The MITRE Corporation
Alice M. Mulvehill	The MITRE Corporation
Lorraine F. Normore	Chemical Abstracts Service
Robert N. Parrish	Teledyne Systems Company
Steven P. Rogers	Anacapa Sciences, Inc.
Eric M. Schaffer	Human Performance Associates

Ben Shneiderman	University of Maryland
Susan G. Tammaro	The MITRE Corporation
Nancy S. Tanner	University of Massachusetts
John C. Thomas	IBM Corporation
Herb Weiner	Tektronix, Inc.
R. Don Williams	Texas Instruments

Some of these people have offered specific suggestions. Some have contributed more general comments about the wording or formatting of the guidelines material. But all have shown a serious concern with trying to improve the guidelines and make them more useful to designers of user interface software. Probably not one of these people would agree with all of the guidelines proposed here; in matters of judgment we can seldom achieve unanimity. But where the guidelines seem good, these are people who deserve our thanks.

Several people on this list deserve extra thanks. Drs. Dray, Kessel, Kornfeld, and Williams traveled to Boston last summer to spend some long days in guidelines review. Our colleagues at MITRE have met for several weeks this past year as a continuing in-house working group for guidelines review. Special thanks are also due to Dr. Arlene Aucella, who helped prepare last year's guidelines report and helped begin this year's revision.

Critical review and revision of the guidelines must continue. No guideline proposed here is worded so perfectly that it cannot be improved. And the need persists for more illustrative examples, more completely noted exceptions, and probably more references.

Perhaps you can help in this work by proposing new guidelines, or by suggesting improvements to those published here, or by citing examples, exceptions, and references. If so, please copy the change sheet included at the back of this report, and use it to send your suggestions. You should send comments not only if you believe a guideline is wrong, but also if a guideline does not seem clearly stated. If the wording of a guideline is not clear to you, then it will probably confuse other people who try to use it in the future.

Albert, A. E. (1982). The effect of graphic input devices on performance in a cursor positioning task. In Proceedings of the Human Factors Society 26th Annual Meeting (pp. 54-58). Santa Monica, CA: Human Factors Society.

Aretz, A. J., and Kopala, C. J. (1981). Automatic return in multifunction control logic. In Proceedings of the Human Factors Society 25th Annual Meeting (pp. 254-256). Santa Monica, CA: Human Factors Society.

Barnard, P. J., Hammond, N. V., MacLean, A., and Morton, J. (1982). Learning and remembering interactive commands in a text-editing task. Behaviour and Information Technology, 1, 347-358.

Bertelson, P., Boons, J.-P., and Renkin, A. (1965). Vitesse libre et vitesse imposee dans une tache simulant le tri mechanique de la correspondence [Self pacing and imposed pacing in a task simulating automated postal sorting]. Ergonomics, 8, 3-22.

Bertoni, P. (1982, June 22). Of slipped disks . . . . Boston Globe.

Billingsley, P. A. (1982). Navigation through hierarchical menu structures: Does it help to have a map? In Proceedings of the Human Factors Society 26th Annual Meeting (pp. 103-107). Santa Monica, CA: Human Factors Society.

BB = Brown, C. M., Brown, D. B., Burkleo, H. V., Mangelsdorf, J. E., Olsen, R. A., and Perkins, R. D. (1983, June 15). Human Factors Engineering Standards for Information Processing Systems (LMSC-D877141). Sunnyvale, CA: Lockheed Missiles and Space Company.

Bury, K. F., Boyle, J. M., Evey, R. J., and Neal, A. S. (1982). Windowing versus scrolling on a visual display terminal. Human Factors, 24, 385-394.

Butterbaugh, L. C., and Rockwell, T. H. (1982). Evaluation of alternative alphanumeric keying logics. Human Factors, 24, 521-533.

Campbell, A. J., Marchetti, F. M., and Mewhort, D. J. K. (1981). Reading speed and text production: A note on right-justification techniques. Ergonomics, 24, 633-640.

REFERENCES

Carroll, J. M. (1982). Learning, using and designing filenames and command paradigms. Behaviour and Information Technology, 1, 327-346.

Cohill, A. M., and Williges, R. C. (1982). Computer-augmented retrieval of HELP information for novice users. In Proceedings of the Human Factors Society 26th Annual Meeting (pp. 79-82). Santa Monica, CA: Human Factors Society.

Darnell, M. J., and Neal, A. S. (1983). Text editing performance with partial and full page displays. In Proceedings of the Human Factors Society 27th Annual Meeting (pp. 821-825). Santa Monica, CA: Human Factors Society.

Dean, M. (1982). How a computer should talk to people. IBM Systems Journal, 21, 424-453.

Demers, R. A. (1981). System design for usability. Communications of the ACM, 24, 494-501.

Dray, S. M., Ogden, W. G., and Vestewig, R. E. (1981). Measuring performance with a menu-selection human-computer interface. In Proceedings of the Human Factors Society 25th Annual Meeting (pp. 746-748). Santa Monica, CA: Human Factors Society.

Durding, B. M., Becker, C. A., and Gould, J. D. (1977). Data organization. Human Factors, 19, 1-14.

Dwyer, B. (1981). A user-friendly algorithm. Commuications of the ACM, 24, 556-561.

Ehrenreich, S. L. (1981). Query languages: Design recommendations derived from the human factors literature. Human Factors, 23, 709-725.

Ehrenreich, S. L., and Porcu, T. (1982). Abbreviations for automated systems: Teaching operators the rules. In A. Badre and B. Shneiderman (Eds.), Directions in Human/Computer Interaction (pp. 111-135). Norwood, NJ: Ablex Publishing.

Elkerton, J., Williges, R. C., Pittman, J. A., and Roach, J. (1982). Strategies of interactive file search. In Proceedings of the Human Factors Society 26th Annual Meeting (pp. 83-86). Santa Monica, CA: Human Factors Society.

G = Engel, S. E., and Granda, R. E. (1975, December). Guidelines for Man/Display Interfaces (Technical Report TR 00.2720). Poughkeepsie, NY: IBM.

Foley, J. D., and Van Dam, A. (1982). Fundamentals of Interactive Computer Graphics. Reading, MA: Addison-Wesley.

Foley, J. D., and Wallace, V. L. (1974). The art of natural graphic man-machine conversation. Proceedings of the IEEE, 62, 462-471.

Gade, P. A., Fields, A. F., Maisano, R. E., Marshall, C. F., and Alderman, I. N. (1981). Data entry performance as a function of method and instructional strategy. Human Factors, 23, 199-210.

Galitz, W. O. (1980). Human Factors in Office Automation. Atlanta, GA: Life Office Management Association.

Geiser, G., Schumacher, W., and Berger, L. (1982). Talking keyboard for user guidance in multifunction systems. In Proceedings of the Human Factors Society 26th Annual Meeting (pp. 436-439). Santa Monica, CA: Human Factors Society.

Gilfoil, D. M. (1982). Warming up to computers: A study of cognitive and affective interaction over time. In Proceedings of Conference on Human Factors in Computer Systems (pp. 245-250). Washington, DC: Association for Computing Machinery.

Goodwin, N. C. (1974, March). INTRO -- In Which a Smart Terminal Teaches Its Own Use (Technical Report ESD-TR-74-374). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A126 205)

Goodwin, N. C. (1975). Cursor positioning on an electronic display using lightpen, lightgun, or keyboard for three basic tasks. Human Factors, 17, 289-295.

Goodwin, N. C. (1980). A user-oriented evaluation of computer-aided message handling. In Proceedings of the Human Factors Society 24th Annual Meeting (pp. 585-589). Santa Monica, CA: Human Factors Society.

Goodwin, N. C. (1982). Effect of interface design on usability of message handling systems. In Proceedings of the Human Factors Society 26th Annual Meeting (pp. 69-73). Santa Monica, CA: Human Factors Society.

Goodwin, N. C. (1983). Designing a multipurpose menu driven user interface to computer based tools. In Proceedings of the Human Factors Society 27th Annual Meeting (pp. 816-820). Santa Monica, CA: Human Factors Society.

REFERENCES

Goodwin, N. C. (1984). Building a usable office support system from diverse components. In Proceedings of INTERACT'84 Conference on Human-Computer Interaction. Amsterdam: Elsevier Science Publishers.

Gould, J. D. (1981). Composing letters with computer-based text editors. Human Factors, 23, 593-606.

Granda, R. E., Teitelbaum, R. C., and Dunlap, G. L. (1982). The effect of VDT command line location on data entry behavior. In Proceedings of the Human Factors Society 26th Annual Meeting (pp. 621-624). Santa Monica, CA: Human Factors Society.

Gregory, M., and Poulton, E. C. (1970). Even versus uneven right-hand margins and the rate of comprehension in reading. Ergonomics, 13, 427-434.

Hamill, B. W. (1980). Experimental document design: Guidebook organization and index formats. In Proceedings of the Human Factors Society 24th Annual Meeting (pp. 480-483). Santa Monica, CA: Human Factors Society.

Hanson, R. H., Payne, D. G., Shiveley, R. J., and Kantowitz, B. H. (1981). Process control simulation research in monitoring analog and digital displays. In Proceedings of the Human Factors Society 25th Annual Meeting (pp. 154-158). Santa Monica, CA: Human Factors Society.

Hartley, J., Young, M., and Burnhill, P. (1975). On the typing of tables. Applied Ergonomics, 6, 39-42.

Hirsh-Pasek, K., Nudelman, S., and Schneider, M. L. (1982). An experimental evaluation of abbreviation schemes in limited lexicons. Behaviour and Information Technology, 1, 359-369.

Hollingsworth, S. R., and Dray, S. M. (1981). Implications of post-stimulus cueing of response options for the design of function keyboards. In Proceedings of the Human Factors Society 25th Annual Meeting (pp. 263-265). Santa Monica, CA: Human Factors Society.

Human Factors Society. (1984). Author's Guide. Santa Monica, CA: Author.

Keister, R. S., and Gallaway, G. R. (1983). Making software user friendly: An assessment of data entry performance. In Proceedings of the Human Factors Society 27th Annual Meeting (pp. 1031-1034). Santa Monica, CA: Human Factors Society.

Krohn, G. S. (1983). Flowcharts used for procedural instructions. Human Factors, 25, 573-581.

Lee, A., and Lochovsky, F. H. (1983). Enhancing the usability of an office information system through direct manipulation. In Proceedings of CHI'83 Human Factors in Computing Systems (pp. 130-134). New York: Association for Computing Machinery.

Liebelt, L. S., McDonald, J. E., Stone, J. D., and Karat, J. (1982). The effect of organization on learning menu access. In Proceedings of the Human Factors Society 26th Annual Meeting (pp. 546-550). Santa Monica, CA: Human Factors Society.

Limanowski, J. J. (1983). On-line documentation systems: History and issues. In Proceedings of the Human Factors Society 27th Annual Meeting (pp. 1027-1030). Santa Monica, CA: Human Factors Society.

Magers, C. S. (1983). An experimental evaluation of on-line HELP for non-programmers. In Proceedings of CHI'83 Human Factors in Computing Systems (pp. 277-281). New York: Association for Computing Machinery.

Martin, J. (1973). Design of Man-Computer Dialogues. Englewood Cliffs, NJ: Prentice-Hall.

Martin, J. (1978). The Wired Society. Englewood Cliffs, NJ: Prentice-Hall.

McDonald, J. E., Stone, J. D., and Liebelt, L. S. (1983). Searching for items in menus: The effects of organization and type of target. In Proceedings of the Human Factors Society 27th Annual Meeting (834-837). Santa Monica, CA: Human Factors Society.

Michard, A. (1982). Graphical presentation of boolean expressions in a database query language: Design notes and an ergonomic evaluation. Behaviour and Information Technology, 1, 279-288.

MIL-H-48655B. (1979, 31 January). Military Specification: Human Engineering Requirements for Military Systems, Equipment and Facilities. Washington, DC: Department of Defense.

REFERENCES

MIL-STD-12D. (1981, 29 May). Abbreviations for Use on Drawings, and in Specifications, Standards and Technical Documents. Washington, DC: Department of Defense.

MIL-STD-1472B. (1974, 31 December). Military Standard: Human Engineering Design Criteria for Military Systems, Equipment and Facilities. Washington, DC: Department of Defense.

MS = MIL-STD-1472C, Revised. (1983, 1 September). Military Standard: Human Engineering Design Criteria for Military Systems, Equipment and Facilities. Washington, DC: Department of Defense.

Miller, D. P. (1981). The depth/breadth tradeoff in hierarchical computer menus. In Proceedings of the Human Factors Society 25th Annual Meeting (pp. 296-300). Santa Monica, CA: Human Factors Society.

Miller, R. B. (1968). Response time in user-system conversational transactions. In Proceedings of the AFIPS Fall Joint Computer Conference, 33, 267-277.

Morrill, C. S. (1967). Computer-aided instruction as part of a management information system. Human Factors, 9, 251-256.

Morrill, C. S., and Davies, B. L. (1961). Target tracking and acquisition in three dimensions using a two-dimensional display surface. Journal of Applied Psychology, 45, 214-221.

Morrill, C. S., Goodwin, N. C., and Smith, S. L. (1968). User input mode and computer-aided instruction. Human Factors, 10, 225-232.

Moses, F. L., and Ehrenreich, S. L. (1981). Abbreviations for automated systems. In Proceedings of the Human Factors Society 25th Annual Meeting (pp. 132-135). Santa Monica, CA: Human Factors Society.

NASA (National Aeronautics and Space Administration). (1979, January). Spacelab Experiment Computer Application Software (ECAS) Display Design and Command Usage Guidelines (Report MSFC-PROC-711). George C. Marshall Space Flight Center, AL: Author.

Neal, A. S., and Emmons, W. H. (1982). Operator corrections during text entry with word processing systems. In Proceedings of the Human Factors Society 26th Annual Meeting (pp. 625-628). Santa Monica, CA: Human Factors Society.

Nickerson, R. S., and Pew, R. W. (1971, June). Oblique steps toward the human-factors engineering of interactive computer systems. Published as an Appendix in M. C. Grignetti, D. C. Miller, R. S. Nickerson and R. W. Pew, Information Processing Models and Computer Aids for Human Performance (Report No. 2190). Cambridge, MA: Bolt, Beranek and Newman. (NTIS No.AD A732 913)

Noyes, L. (1980). The positioning of type on maps: The effect of surrounding material on word recognition. Human Factors, 22, 353-360.

Palme, J. (1979). A human-computer interface for non-computer specialists. Software -- Practice and Experience, 9, 741-747.

Parrish, R. N., Gates, J. L., Munger, S. J., Grimma, P. R., and Smith, L. T. (1982, February). Development of Design Guidelines and Criteria for User/Operator Transactions with Battlefield Automated Systems, Phase II Final Report: Volume II, Prototype Design Handbook for Combat and Materiel Developers (Report WF-80-AE-00). Alexandria, VA: US Army Research Institute.

Parsons, H. M. (1970). The scope of human factors in computer-based data processing systems. Human Factors, 12, 165-175.

Penniman, W. D. (1979, May). Past chairman's message. SIG Newsletter No. UOI-10. Washington, DC: American Society for Information Science.

PR = Pew, R. W., and Rollins, A. M. (1975). Dialog Specification Procedures (Report 3129, revised). Cambridge, MA: Bolt Beranek and Newman.

Ramsey, H. R., and Atwood, M. E. (1979, September). Human Factors in Computer Systems: A Review of the Literature (Technical Report SAI-79-111-DEN). Englewood, CO: Science Applications, Inc. (NTIS No. AD A075 679)

Ramsey, H. R., and Atwood, M. E. (1980). Man-computer interface design guidance: State of the art. In Proceedings of the Human Factors Society 24th Annual Meeting (pp. 85-89). Santa Monica, CA: Human Factors Society.

REFERENCES

Ramsey, H. R., Atwood, M. E., and Kirshbaum, P. J. (1978, May). A Critically Annotated Bibliography of the Literature of Human Factors in Computer Systems (Technical Report SAI-78-070-DEN). Englewood, CO: Science Applications, Inc. (NTIS No. AD A058 081)

Reisner, P. (1977). Use of psychological experimentation as an aid to development of a query language. IEEE Transactions on Software Engineering, SE-3, 218-229.

Reisner, P. (1981). Formal grammar and human factors design of an interactive graphics system. IEEE Transactions on Software Engineering, SE-7, 229-240.

Roberts, T. L., and Moran, T. P. (1983). The evaluation of text editors: methodology and empirical results. Communications of the ACM, 26, 265-283.

Savage, R. E., Habinek, J. K., and Blackstad, N. J. (1982). An experimental evaluation of input field and cursor combinations. In Proceedings of the Human Factors Society 26th Annual Meeting (pp. 629-633). Santa Monica, CA: Human Factors Society.

Seibel, R. (1972). Data entry devices and procedures. In H. P. Van Cott and R. G. Kinkade (Eds.), Human Engineering Guide to Equipment Design (pp. 311-344). Washington, DC: U. S. Government Printing Office.

Shneiderman, B. (1980). Software Psychology: Human Factors in Computer and Information Systems. Cambridge, MA: Winthrop Publishers.

Shneiderman, B. (1981). A note on human factors issues of natural language interaction with database systems. Information Systems, 6, 125-129.

Shneiderman, B. (1982). The future of interactive systems and the emergence of direct manipulation. Behaviour and Information Technology, 1, 237-256.

Sidorsky, R. C., and Parrish, R. N. (1980). Guidelines and criteria for human-computer interface design of battlefield automated systems. In Proceedings of the Human Factors Society 24th Annual Meeting (pp. 98-102). Santa Monica, CA: Human Factors Society.

Smith, S. L. (1962a). Angular estimation. Journal of Applied Psychology, 46, 240-246.

Smith, S. L. (1962b). Color coding and visual search. Journal of Experimental Psychology, 64, 434-440.

Smith, S. L. (1963a). Color coding and visual separability in information displays. Journal of Applied Psychology, 47, 358-364. (a)

Smith, S. L. (1963b). Man-computer information transfer. In J. H. Howard (Ed.), Electronic Information Display Systems (pp. 284-299). Washington, DC: Spartan Books.

Smith, S. L. (1980, June). Requirements definition and design guidelines for the man-machine interface in C3 system acquisition (Technical Report ESD-TR-80-122). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A087 258)

Smith, S. L. (1981a, February). Man-Machine Interface (MMI) Requirements Definition and Design Guidelines: A Progress Report (Technical Report ESD-TR-81-113). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A096 705)

Smith, S. L. (1981b). Exploring compatibility with words and pictures. Human Factors, 23, 305-315.

Smith, S. L. (1982a). "User-system interface". Human Factors Society Bulletin, 25(3), 1.

Smith, S. L. (1982b, April). User-System Interface Design for Computer-Based Information Systems (Technical Report ESD-TR-82-132). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A115 853)

Smith, S. L. (1984, April). User-System Interface Design in System Acquisition (Technical Report ESD-TR-84-158). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A140 956)

Smith, S. L., and Aucella, A. F. (1983a, March). Design Guidelines for the User Interface to Computer-Based Information Systems (Technical Report ESD-TR-83-122). Hanscom Air Force Base, MA: USAF Electronic Systems Division. (NTIS No. AD A127 345)

REFERENCES

Smith, S. L., and Aucella, A. F. (1983b). Numbering formats for hierarchic lists. Human Factors, 25, 343-348.

Smith, S. L., Farquhar, B. B., and Thomas, D. W. (1965). Color coding in formatted displays. Journal of Applied Psychology, 49, 393-398.

Smith, S. L., and Goodwin, N. C. (1970). Computer-generated speech and man-computer interaction. Human Factors, 12, 215-223.

Smith, S. L., and Goodwin, N. C. (1971a). Alphabetic data entry via the Touch-Tone pad: A comment. Human Factors, 13, 189-190.

Smith, S. L., and Goodwin, N. C. (1971b). Blink coding for information display. Human Factors, 13, 283-290.

Smith, S. L., and Goodwin, N. C. (1972). Another look at blinking displays. Human Factors, 14, 345-347.

Smith, S. L., and Mosier, J. N. (1984). The user interface to computer-based information systems: A survey of current software design practice. Behaviour and Information Technology (in press).

Smith, S. L., and Thomas, D. W. (1964). Color versus shape coding in information displays. Journal of Applied Psychology, 48, 137-146.

Snowberry, K., Parkinson, S. R., and Sisson, N. (1983). Computer display menus. Ergonomics, 26, 699-712.

Stewart, T. (1980). Communicating with dialogues. Ergonomics, 23, 909-919.

Symons, C. R., and Schweitzer, J. A. (1984). A proposal for an automated access control standard. ACM/SIGCHI Bulletin, 16(1), 17-23.

Tammaro, S. G. (1983). An analysis of the use of computer-based office support tools: User characteristics and usage patterns. In Proceedings of the Human Factors Society 27th Annual Meeting (pp. 882-886). Santa Monica, CA: Human Factors Society.

Thompson, D. A. (1971). Interface design for an interactive information retrieval system: a literature survey and a research system description. Journal of the American Society for Information Science, 22, 361-373.

Tullis, T. S. (1981). An evaluation of alphanumeric, graphic, and color information displays. Human Factors, 23, 541-550.

Whalley, P. C., and Fleming, R. W. (1975). An experiment with a simple recorder of reading behaviour. Programmed Learning and Educational Technology, 12, 120-124.

Whitfield, D., Ball R. G., and Bird, J. M. (1983). Some comparisons of on-display and off-display touch input devices for interaction with computer generated displays. Ergonomics, 26, 1033-1053.

Woodson, W. E. (1981). Human Factors Design Handbook. New York: McGraw-Hill.

Wright, P., and Lickorish, A. (1983). Proof-reading texts on screen and paper. Behaviour and Information Technology, 2, 227-236.

Wright, P., and Reid, F. (1973). Written information: Some alternatives to prose for expressing the outcomes of complex contingencies. Journal of Applied Psychology, 57, 160-166.

GUIDELINE TITLES

In this report, the guidelines material has grown too bulky for easy scanning. If a reader wants to find guidelines pertaining to a particular subject, some more efficient means is needed to find specific material.

The report begins with a table of contents, but that shows only the general functional organization of the guidelines material. Section titles are given, but there is no information about specific topical coverage within sections.

The report ends with a topical index. But any index, no matter how carefully prepared, will sometimes fail to direct us to the material we seek. Sometimes a seeker's words to describe a particular topic will be different from the words chosen by the person who constructed the index.

As an intermediate expedient, more detailed than a table of contents but more structured than an index, listed here are the titles of all 679 guidelines. This listing provides a relatively compact summary of topical coverage within different sections of the guidelines material, in words actually used in the guidelines. A reader who is not already familiar with the guidelines may find these titles helpful for locating specific information.

Such a listing of guidelines might also be used as the basis of a checklist for evaluating the design of user interface software. The use of guidelines for design evaluation will be discussed at greater length in another report.

DATA ENTRY	General	Guideline Number 1.0
• Entry via primary display		-1
• Feedback during data entry		-2
o Fast response		-3
• Single method for entering data		-4
• Defined display areas for data entry		-5
• Consistent method for data change		-6
• User-paced data entry		-7
• Explicit ENTER action		-8
o ENTER key labeling		-9
• Explicit CANCEL action		-10
• Feedback for completion of data entry		-11
o Feedback for repetitive data entries		-12
o Feedback when changing data		-13
• Keeping data items short		-14
o Partitioning long data items		-15
• Optional abbreviation		-16
o Distinctive abbreviation		-17
o Consistent abbreviation rule		-18
o Minimal exceptions to abbreviation rule		-19
o Minimal deviation from abbreviation rule		-20
o Fixed abbreviation length		-21
o Clarifying unrecognized abbreviations		-22
• Prompting data entry		-23
• Character entry via single keystroke		-24
o Minimal shift keying		-25
• Upper/lower case equivalent		-26
• Decimal point optional		-27
• Leading zeros optional		-28
• Single/multiple blanks equivalent		-29

DATA ENTRY	Position Designation	1.1

• Distinctive cursor	-1
o Non-obscuring cursor	-2
o Precise pointing	-3
• Explicit activation	-4
• Fast Response	-5
• Stable cursor	-6
• Responsive cursor control	-7
• Consistent incremental positioning	-8
o Variable step size	-9
o Proportional spacing	-10
• Continuous cursor positioning	-11
• Direct pointing	-12
o Large pointing area for option selection	-13
• Cursor control at keyboard	-14
• Compatible control of cursor movement	-15
• Minimal use of multiple cursors	-16
o Distinctive multiple cursors	-17
o Distinctive control of multiple cursors	-18
o Compatible control of multiple cursors	-19
• Consistent HOME position	-20
• Consistent cursor placement	-21
• Easy cursor movement to data fields	-22
• Display format protection	-23
• Data entry independent of cursor placement	-24

DATA ENTRY	Direction Designation	1.2

• Analog entry of estimated direction	-1
• Keyed entry of quantified direction	-2

SEQUENCE CONTROL — Function Keys — 3.1.4

SEQUENCE CONTROL — Menu Selection — 3.1.3

SEQUENCE CONTROL	General	3.0

- Flexible sequence control -1
- Minimal user actions -2
- Control matched to user skill -3
- User initiative in sequence control -4
- Control by explicit user action -5

- Consistent user actions -6
- Logical transaction sequences -7
- Distinctive display of control information -8
- Displayed context -9
- Consistent terminology for sequence control -10

- Feedback for control entries -11
  - o Indicating completion of processing -12
  - o Compatibility with user expectations -13
- User-paced sequence control -14
- Appropriate computer response time -15

- Control availability -16
  - o Indicating control lockout -17
  - o Interrupt to end control lockout -18
- Control by simultaneous users -19

SEQUENCE CONTROL	Dialogue Type	3.1

- Dialogue matched to user and task -1
- Appropriate computer response time -2

SEQUENCE CONTROL	Question and Answer	3.1.1

- Question-and-answer dialogue -1
- Questions displayed singly -2
- Recapitulating prior answers -3
- Sequence compatible with source documents -4

SEQUENCE CONTROL	Form Filling	3.1.2

- Form filling for data entry -1
- Form filling for control entry -2
- Defaults for control entry -3
- Consistent format for control forms -4

DATA DISPLAY	Suppression	2.8

- Temporary suppression of displayed data -1
- Labeling display suppression -2
- Signaling changes to suppressed data -3
- Resuming display of suppressed data -4

DATA DISPLAY	Design Change	2.9

- Flexible design for data display -1

DATA DISPLAY — Coding 2.4

DATA DISPLAY	Format	2.3

DATA DISPLAY - Data Type	Tables	2.1.3

• Tables for data comparison	-1
• Column and row labels	-2
o Labeling units of measurement	-3
• Justification of numeric data	-4
o Justification of alphabetic data	-5
• Logical organization	-6
o Tables referenced by first column	-7
o Items paired for direct comparison	-8
• Distinctive labeling	-9
o Numbered items start with "1"	-10
o Repeated elements in hierarchic numbering	-11
• Row scanning cues	-12
o Column scanning cues	-13
o Consistent column spacing	-14
• Consistent label format	-15

DATA DISPLAY - Data Type	Graphics	2.1.4

• Graphic display for data comparison	-1
o Graphic displays for monitoring data change	-2
• Conventional flowchart orientation	-3
• Standarized graphics symbology	-4

DATA DISPLAY - Data Type	Combination	2.1.5

• Mixing text with figures	-1

DATA DISPLAY	Density	2.2

• Necessary data displayed	-1
o Only necessary data displayed	-2

DATA DISPLAY - Data Type — Text	2.1.1

DATA DISPLAY - Data Type — Data Forms	2.1.2

DATA DISPLAY	General	2.0

• Necessary data displayed	-1
o Only necessary data displayed	-2
• Data displayed in usable form	-3
• Data display consistent with user conventions	-4
o Establishing display standards	-5
• Consistent display format	-6
• User control of data display	-7
o User changes to displayed data	-8
o Protection of displayed data	-9
• Context for displayed data	-10
• Familiar wording	-11
o Consistent wording	-12
o Consistent wording across displays	-13
• Minimal use of abbreviation	-14
o Consistent abbreviation	-15
o Distinctive abbreviations	-16
o Dictionary of abbreviations	-17
o Minimal punctuation of abbreviations	-18

DATA DISPLAY	Data Type	2.1

• Appropriate data types	-1

DATA ENTRY	Design Change	1.9

- Flexible design for data entry -1

DATA ENTRY	Tables	1.5

- Tables for related data sets -1
- Distinctive labels -2
- o Informative labels -3
- Tabbing within rows -4
- o Tabbing within columns -5

- Automatic justification of entries -6
- o Justification of numeric entries -7
- Aiding entry of duplicative data -8
- Row scanning cues -9

DATA ENTRY	Graphics	1.6

(No entries)

DATA ENTRY	Data Validation	1.7

- Automatic data validation -1
- Accepting correct entries -2
- *Non-disruptive error messages* -3
- Deferral of required data entry -4
- o Reminder of deferred entry -5

- Timely validation of sequential transactions -6
- Optional item-by-item validation -7

DATA ENTRY	Other Data Processing	1.8

- Default values -1
- o User definition of default values -2
- o Display of default values -3
- o Easy confirmation to enter default values -4
- o Temporary replacement of default values -5

- Automatic generation of routine data -6
- Automatic computation of derived data -7
- User review of prior entries -8
- Automatic entry of redundant data -9
- Automatic cross-file updating -10

DATA ENTRY	Data Forms	1.4

DATA ENTRY	Text	1.3

- • Adequate display capacity -1
- • Full editing capabilities during text entry -2
- • Free cursor movement -3
- o Control entries distinct from text -4
- • Natural units of text -5

- o Control entry based on units of text -6
- o Highlighting specified text -7
- o Cursor movement by units of text -8
- • String search -9
- o Upper/lower case equivalent in search -10

- o Specifying case in search -11
- o Global search and replace -12
- o Case in global search and replace -13
- • Automatic pagination aids -14
- o User control of pagination -15

- o Controlling integrity of text units -16
- • Automatic line break -17
- o Consistent word spacing -18
- o Hyphenation by users -19
- • Format control by user -20

- • Establishing predefined formats -21
- o Storing user-defined formats -22
- • Moving text -23
- • Storing frequently used text -24
- • Necessary data displayed -25

- o Text distinct from annotation -26
- • Printing for proofreading -27
- • Text displayed as printed -28
- • Flexible printing options -29
- • Information on printing status -30

- • Auditory signals for alerting users -31
- • Protecting text during page overruns -32
- • Confirming actions in DELETE mode -33
- • Reversible actions -34
- • User confirmation of editing changes -35

SEQUENCE CONTROL — Command Language — 3.1.5

- Command language -1
- Standard display area for command entry -2
- Functional command language design -3
- Layered command language -4
- Familiar wording -5

o Consistent wording of commands -6
o Distinctive wording of commands -7
- User-assigned command names -8
- User-requested prompts -9
o General list of commands -10

- Command stacking -11
o User definition of macro commands -12
- Minimal command punctuation -13
o Standard command delimiter -14
- Ignoring blanks in command entry -15

- Abbreviation of commands -16
- Standard techniques for command editing -17
o Interpreting misspelled commands -18
- Correcting command entry errors -19
o Aborting erroneous commands -20

- Reviewing destructive commands -21

SEQUENCE CONTROL — Query Language — 3.1.6

- Query language -1
- Natural organization of data -2
o Coherent representation of data organization -3
- Task-oriented wording -4
- Flexible query formulation -5

- Minimal need for quantifiers -6
- Logic to link queries -7
- Linking sequential queries -8
- Confirming large-scale retrieval -9

SEQUENCE CONTROL — Natural Language — 3.1.7

o Constrained natural language -1

SEQUENCE CONTROL	Context Definition	3.4

• Defining context for users -1
o Context established by prior entries -2
o Record of prior entries -3
• Display of operational mode -4
• Display of control parameters -5

• Highlighting selected data -6
• Consistent display of context information -7

SEQUENCE CONTROL	Error Management	3.5

• Appropriate response to all entries -1
• Command editing -2
• Prompting command correction -3
• Errors in stacked commands -4
o Partial execution of stacked commands -5

• Explicit entry of corrections -6
• User confirmation of destructive entries -7
o User warned of potential data loss -8
o Distinctive *CONFIRM action* -9
• UNDO to reverse control actions -10

o Preventing data loss at LOG-OFF -11
o Immediate data correction -12
o Flexible BACKUP for error correction -13

SEQUENCE CONTROL	Alarms	3.6

• Alarm definition by user -1
• Distinctive and consistent alarms -2
• Alarm acknowledgement -3
o Alarm reset -4
o Acknowledgement of critical alarms -5

SEQUENCE CONTROL	Design Change	3.7

• Flexible design for sequence control -1

USER GUIDANCE	General	4.0

USER GUIDANCE	Status Information	4.1

USER GUIDANCE	Routine Feedback	4.2
• Consistent feedback		-1
• Fast response		-2
• Feedback for control entries		-3
o Indicating completion of processing		-4
• Feedback for print requests		-5
• Display identification		-6
o Identifying multipage displays		-7
• Indicating operational mode		-8
o Indicating option selection		-9
o Indicating item selection		-10
• Feedback for user interrupt		-11

USER GUIDANCE	Error Feedback	4.3
• Informative error messages		-1
o Specific error messages		-2
o Task-oriented error messages		-3
• Advisory error messages		-4
• Brief error messages		-5
• Neutral wording for error messages		-6
• Multilevel error messages		-7
• Multiple error messages		-8
o Indicating repeated errors		-9
• Non-disruptive error messages		-10
• Appropriate response time for error messages		-11
• Documenting error messages		-12
• Cursor placement following error		-13
• User editing of entry errors		-14
• Cautionary messages		-15
• User confirmation of destructive entries		-16
• Alarm coding		-17

USER GUIDANCE	Job Aids	4.4

USER GUIDANCE	User Records	4.5

USER GUIDANCE	Design Change	4.6

- Flexible design for user guidance -1
- Notifying users of design changes -2

DATA TRANSMISSION	General	5.0

- Consistent procedures -1
- Minimal user actions -2
- Minimal memory load on user -3
- Message composition compatible with data entry -4

o Message viewing compatible with data display -5

- Flexible user control -6
- Control by explicit user action -7

DATA TRANSMISSION	Data Type	5.1

- User-designed formats -1
- Automatic text formatting -2
- Unformatted text -3
- Data forms -4

o Tables and graphics -5

- Message highlighting -6

DATA TRANSMISSION	Sending	5.2

- Source selection -1
- Destination selection -2
- Status information -3
- Assignment of priority -4
- Message printing -5

DATA TRANSMISSION	Receiving	5.3

- Source selection -1
- Destination selection -2

o Receipt by priority -3

DATA TRANSMISSION	Transmission Control	5.4

- Functional wording -1
- Flexible data specification -2
- Automatic message formatting -3
- Automatic message routing -4
- Automatic message initiation -5

- User review of transmitted data -6

DATA TRANSMISSION	Feedback	5.5

- Automatic feedback -1
- Feedback for message sent -2
- Information about messages received -3
- User specification of feedback -4

DATA TRANSMISSION	Queuing	5.6

- Automatic queuing -1
- Deferring message transmission -2
o Queuing failed transmissions -3
- Queuing messages received -4
- Non-disruptive message receipt -5

o Priority indicating for messages received -6
- User review of messages received -7

DATA TRANSMISSION	Record Keeping	5.7

- Automatic record keeping -1

DATA TRANSMISSION	Design Change	5.8

- Flexible design for data transmission -1

DATA PROTECTION	General	6.0

- Automatic security measures -1
- Consistent procedures -2
- Control by explicit user action -3
- Feedback for mode selection -4
- Appropriate response to all entries -5

- User review and editing of entries -6
- Resolving ambiguous entries -7
- Users warned of threats to security -8

DATA PROTECTION	User Identification	6.1

- Easy LOG-ON -1
o Prompting LOG-ON -2
- User choice of passwords -3
o Private entry of passwords -4
- Continuous recognition of user identity -5

DATA PROTECTION	Data Access	6.2

- Single authorization for data access -1
- Displayed security classification -2
- User editing of displayed data -3
- Protecting displayed data -4
o Protecting display formats -5

- Easy display suppression -6
- Printing protected data -7
- Automatic records of data access -8

DATA PROTECTION	Data Entry/Change	6.3

- Single authorization for data entry/change -1
- Data entry/change transaction records -2
- Protection from data change -3
- Segregating real from simulated data -4
- Emphasizing accuracy -5

- Simple procedures -6
o Displaying default values -7
- Explicit user actions -8
o Single entry of related data -9
- User editing of data before entry -10

o Immediate error correction -11
o User editing of entry errors -12
- Flexible BACKUP for error correction -13
- Explicit entry of corrections -14
- Data verification by user review -15

o Automatic data generation -16
- Validation of changed data -17
o Cross validation of changed data -18
- Displaying data to be changed -19
- User confirmation of destructive actions -20

DATA PROTECTION	Data Transmission	6.4

- Automatic protection of transmitted data -1
- User review of data before transmission -2
- Saving sent data until receipt is confirmed -3
- Queuing received messages -4
- Printing messages -5

GLOSSARY

A glossary of words dealing with the user interface to computer systems could contain hundreds of terms. Such a glossary would be difficult to compile because terms are often used inconsistently. System analysts, or software designers, will sometimes understand a term differently than a human factors specialist would. The glossary presented here is not intended to be an exhaustive reference for user-system interface terms. Instead, it simply attempts to clarify the meanings of some of the terms used in this report. A term is included in this glossary only if it is used inconsistently in the general literature, or if its meaning in this report is more narrow than that commonly used.

ABORT - A capability that cancels all user entries in a defined transaction sequence.

BACKUP - A capability that returns a user to the last previous display in a defined transaction sequence.

CANCEL - A capability that regenerates (or re-initializes) the current display without processing any entries or changes made by the user.

Category - A grouping of data values along a dimension for operational purposes. For example, an air traffic controller might be instructed to implement the same procedures for all aircraft with speeds in the category of 600 to 800 knots. See also "Value".

Command Language - A type of dialogue in which a user composes control entries with minimal prompting by the computer.

Context Definition - Displaying an indication of previous user actions or computer processing that will affect the results of current actions, in order to help a user predict how the system will respond.

Control Entry - User input for controlling a transaction sequence, such as function key activation, menu selection, command entry, etc.

Cursor - A marker on the display screen that indicates the current position for attention, which may designate a displayed item. The cursor may be positioned under computer control or by the user.

Data - The raw materials from which a user extracts information. Data may include numbers, words, pictures, etc.

Data Base - A collection of data that is stored in the computer for relatively long periods of time.

Data Display - Output of data from a computer to its users. Generally, this phrase denotes visual output, but it may be qualified to indicate a different modality, such as an "auditory display".

Data Entry - User input of data for computer processing, and computer responses to such inputs.

Data Field - An area of the display screen reserved for user entry of a data item.

Data Field Label - An area of the display screen that serves as a prompt for entering a data item. It usually cannot be changed by a user.

Data Item - A set of characters of fixed or variable length that forms a single unit of data. Examples of a data item might be a person's last name, or a ZIP code. Sometimes a data item may contain only a single character. Data items may be entered by a user or may be supplied by the computer.

Data Protection - Functional capabilities that guard against unauthorized data access and tampering, user errors and computer failure.

Data Transmission - Message exchange among system users, and also message exchange with other systems. Transmitted data may include numbers, words, pictures, etc.

Data Validation - Functional capabilities that check data entry items for correct content or format, as defined by software logic.

Default Value - A predetermined, frequently used, value for a data or control entry, intended to reduce required user entry actions.

Dialogue - A structured series of interchanges between a user and a computer terminal. Dialogues can be computer-initiated, e.g., question and answer, or user-initiated, e.g., command language.

F

E

A

B

Windowing - An orientation for display framing in which the user conceives of the display frame as a window moving over a fixed array of data. The opposite of scrolling.

Work Station - The general physical environment in which the user works. It includes such things as computer terminals, source documents, desks, chairs, and lighting.

Suspense File - A temporary collection of data saved by the computer for later use.

System - See "Information System".

Task - A series of transactions that comprises part of a user's defined job.

Terminal - An input/output device used to enter and display data. Data are usually entered via a keyboard, and are usually displayed via a video screen ("soft copy") or a printer ("hard copy").

Text Entry - Initial entry and subsequent editing of textual material, including messages.

Transaction - An action by a user followed by a response from the computer. Transaction is used here to represent the smallest functional "molecule" of user-system interaction.

Transmission Control - Controlling the sequence, content, format, routine, timing, etc., of data transmissions.

Update - See "Display Update".

User - Any person who uses an information system in performing his/her job.

User Guidance - Computer prompts and feedback that aid users in performing their tasks. Examples include data field labels, alarm or alert signals, error messages, and HELP messages.

User Records - Automatic recording of user performance, e.g., data access records, error records, HELP access records.

User-System Interface - All aspects of information system design that affect a user's participation in information handling transactions.

Value - Specific data for a particular dimension or variable. For example, values for an aircraft's speed might be 800 knots during one observation and 500 knots during another. See also "Category".

Variable - See "Dimension".

Menu Selection - A type of dialogue in which the user selects one item out of a list of displayed alternatives, whether the selection is by pointing, by entry of an associated option code, or by activation of an adjacent function key.

Message - Data that are transmitted from one computer user to another as a discrete transaction. See "Data Transmission".

Natural Language - A type of dialogue in which users compose control entries in their natural language, e.g., English, Spanish, French.

Operator - See "User".

Output - See "Data Display".

Page - The data appearing at one time on a single display screen.

Position Designation - User selection and entry of a position on a display, or of a displayed item. See also "Cursor".

Query Language - A type of dialogue in which users compose control entries for displaying specified data from a data base.

Question and Answer - A type of dialogue in which the computer displays questions, one at a time, for a user to answer.

RESTART - A capability that returns a user to the first display in a defined transaction sequence.

Screen - See "Page".

Scrolling - An orientation for display framing in which the user conceives of data as moving behind a fixed display frame. The opposite of windowing.

Sequence Control - Logic and means by which user actions and computer responses are linked to become coherent transactions.

Status Information - Information on current data processing status which is either automatically displayed or displayed to the user upon request, e.g., system load, keyboard lock, LOG-ON or processing delay.

Suppression - User control of display coverage by temporary deletion of specified data categories.

Function - A computer-supported capability provided to users as an aid for task performance. Examples of functions are position designation or direction designation.

Function Key - A key whose activation will effect a control entry.

Generation - See "Display Generation".

Hard Copy - A printed paper display output by the computer.

HELP - A capability that displays information upon user request for on-line guidance. HELP may inform a user generally about system capabilities, or may provide more specific guidance in data handling transactions.

Highlighting - Emphasizing displayed data or format features in some way, e.g., through the use of underlining, bolding, or inverse video.

Information - Organized data that users need to successfully perform their tasks. Information serves as an answer to a user's question about data. It is used here to refer to the effective assimilation of data by a user.

Information System - A computer-supported, task-oriented tool designed to help users perform defined information handling tasks.

Input - See "Control Entry" and "Data Entry".

Interaction - See "Transaction".

Interface - See "User-System Interface".

Interrupt - Stopping an ongoing transaction in order to redirect the course of the processing. Examples of interrupt options are CANCEL, BACKUP, RESTART, ABORT.

Item - See "Data Item".

Job - A set of responsibilites and activities that are defined for each system user.

Label - A title or descriptor that helps a user identify displayed data. See "Data Field Label".

Dimension - A scale or categorization along which data may vary, taking different values at different times. For example, relevant dimensions for an aircraft might include its heading, speed, and altitude. See also "Variable".

Direction Designation - User entry of directional data (azimuth, bearing, heading) on a display.

Display - See "Data Display".

Display Format - The organization of different types of data in a display, including information about the data such as labels, and other user guidance such as prompts, error messages, etc.

Display Framing - User control of display coverage by display movement, including paging, scrolling, offset, expansion, etc.

Display Generation - The means of specification of data outputs, either by a user or automatically by the computer.

Display Tailoring - Designing displays to meet the specific task needs of a user, rather than providing a general display which can be used for many purposes.

Display Update - Regeneration of changed data to show current status, by user request or automatically by the computer.

ENTER - An explicit user action that effects computer processing of user entries. For example, after typing a series of numbers, a user might press an ENTER key that will add them to a data base, subject to data validation.

Entry - See "Data Entry" or "Control Entry".

Error Management - Interface design to facilitate the detection and correction of user errors.

Field - See "Data Field".

File - A collection of data, treated as a single unit, that is stored in the computer.

Format - See "Display Format".

Form Filling - A type of dialogue in which the computer displays forms containing labeled fields for data entry by a user.

Framing - See "Display Framing".

N

O

P

T

U

V

USI Design Guidelines -- Changes/Additions

Guideline number (if referring to one in the report): _____

Proposed wording for guideline: _____

_____

_____

_____

_____

Example: _____

_____

_____

_____

Exception: _____

_____

_____

_____

Comment: _____

_____

_____

_____

References to previously published recommendations or supporting data:

_____

_____

_____

Please mail to:
Sidney L. Smith, A134
The MITRE Corporation
Bedford, MA 01730 USA

Your name: _____

Date: _____